

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</small> PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 14-12-2005		2. REPORT TYPE Final Report		3. DATES COVERED (From – To) 14 June 2004 - 14-Dec-05	
4. TITLE AND SUBTITLE Advanced Agent Methods in an Adversarial Environment				5a. CONTRACT NUMBER FA8655-04-1-3044	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Dr. Michal Pechoucek				5d. PROJECT NUMBER	
				5d. TASK NUMBER	
				5e. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Czech Technical University CTU, FEE-K333 Technicka 2 Prague 6 166 27 Czech Republic				8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0014				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) Grant 04-3044	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <p>The suggested research effort aims at continuation of the research in the field of social knowledge, social behavior and meta-reasoning in multi-agent systems. The main and fundamental theme of this research project is extending meta-reasoning and social knowledge manipulation to the problem of intent modeling in the non-trivial domains of non-cooperative, competitive, i.e. adversarial, behavior. The main ambition is to extend meta-reasoning for the problem of intent modeling in the non-cooperative, cooperative, i.e. primarily adversarial, domain. This effort retains the focus on environments with partial communication inaccessibility. This extension of the metareasoning to adversarial agent behavior is a natural step in applicability of this technology to Information Operations. Dr. Pechoucek has a long and successful history with AFRL, working under grants to Czech Tech University with Dr. Vladimir Marik.</p> <p>Deliverable 1: (month 6) Interim report defining the concept of non-collaboratively and hostility in multi-agent systems, with design of a scenario for investigating agents social behaviour in non-collaborative and adversarial environment.</p> <p>Deliverable 2: (month 12) Interim report describing potential role of agent's acquaintance model, social knowledge and meta-reasoning in non-collaborative environment with partial communication accessibility.</p> <p>Deliverable 3: (month 18) Comprehensive final report providing technical description of the entire research effort including formal definition of measures and quantities of agents collective behaviour, adapted social knowledge model, meta-reasoning model and coalition formation algorithms for non-collaborative and adversarial environments and proof-of-concept experimental verification.</p>					
15. SUBJECT TERMS EOARD, Command and Control, Agent Based Systems, Artificial Intelligence					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UL	18. NUMBER OF PAGES 134	19a. NAME OF RESPONSIBLE PERSON PAUL LOSIEWICZ, Ph. D.
a. REPORT UNCLAS	b. ABSTRACT UNCLAS	c. THIS PAGE UNCLAS			19b. TELEPHONE NUMBER (Include area code) +44 20 7514 4474

Advanced Agent Methods in Adversarial Environment

contract number:
FA8655-04-1-3044

Final report, November 30th 2005

Michal Pěchouček^{Pi}, Martin Reháček, David Šišlák,
Petr Benda, Lukáš Foltýn, Jan Tožička, Pavel Jisl, Přemysl Volf



Gerstner Laboratory,
Czech Technical University in Prague
<http://gerstner.felk.cvut.cz>

Contents

1	Introduction – Technical Statement of Work	6
1.1	Introduction	7
2	Formal Model of Adversarial Behavior	9
2.1	Introduction	9
2.2	Adversarial Behavior Definition	10
2.2.1	Properties of Adversarial Action	13
2.3	Examples of Adversarial Behaviour	15
2.3.1	Adversariality in Coalition Formation	15
2.3.2	Adversariality Flight-plan Deconfliction	16
2.4	Conclusions	17
3	Inaccessible Environment and its Properties	19
3.1	Measuring Inaccessibility	19
3.1.1	Agent-to-Agent Accessibility	19
3.1.2	Accessibility Characteristics	20
3.1.3	Accessibility Temporal Characteristics	21
3.2	Measure of Accessibility - Experiments	22
3.2.1	Testing Scenario	22
3.2.2	Results	22
4	Modelling Inaccessibility and Adversarial Behavior	24
4.1	Domain Description – ACROSS	24
4.2	Modelling Communication Inaccessibility	26
4.3	Modelling Adversarial Actions	28
4.4	Scenario Evolution: Humanitarian Relief Operation in Adversarial Environment	29
4.4.1	Inaccessibility and Communication Failures	29
4.4.2	Humanitarian Agents	29
4.4.3	Simulation Agents	31
4.5	Conclusion	32
5	Using Trust for Coalition Formation in Adversarial Environment	33
5.1	State of the art	34
5.2	Requirements on Trust Representation	35
5.3	Fuzzy Numbers	36
5.4	Formal Model	37
5.4.1	Deriving Trust Observations from Coalition Cooperation Results	37

5.4.2	Iterative Learning of Trust Values	38
5.4.3	Self-Trust as a Parameter for Trusting Decisions.....	41
5.4.4	The Decision to Cooperate and Partner Selection.....	41
5.4.5	Trust Module Overview	42
5.5	Experiments	43
5.5.1	Configuration of Experiments	43
5.5.2	Results - Coalitions	43
5.6	Results - Trust	46
5.6.1	Observations.....	49
6	Solving Inaccessibility in the Adversarial Environment.....	51
6.1	Presentation of Inaccessibility Solutions	51
6.1.1	Relay Agents	51
6.1.2	Middle Agents	51
6.1.3	Acquaintance Models	52
6.1.4	Stand-In Agent	52
6.1.5	Solution Selection	54
6.1.6	Experiments	55
6.2	Optimizing the Inaccessibility Solutions	57
6.2.1	Middle Agent Architecture	58
6.2.2	Swarming Controller.....	59
6.2.3	Experiments	61
6.2.4	Information Propagator Adaptation	62
6.2.5	Adaptation to the Changing Environment	63
6.2.6	Observations.....	65
7	Efficient Teamwork in Inaccessible and Adversarial Environment	66
7.1	Problem Statement	66
7.2	Stand-In Agents.....	68
7.3	Trust-Based Planning for Adversarial Domains	69
7.4	Formal Problem Statement	71
7.4.1	Public, Semi-Private and Private Information	72
7.5	Algorithm Description	73
7.5.1	Initial Planning	73
7.5.2	Local Plan Evaluation	75
7.5.3	Coherence & Verification Phase	77
7.5.4	Plan Execution	77
7.6	Algorithm properties.....	77
7.6.1	Overall Characteristics	78
7.6.2	Stability of Flexible & Fuzzy Linear Programming	78
7.7	Conclusions and Future Work	79
8	Conclusions and Future Work	80
A	Progress in \mathcal{A}-globe Multi-Agent Platform Development	82
A.1	System Architecture	82
A.2	Simulation Support in \mathcal{A} -globe	83
A.3	New features and Enhancements in \mathcal{A} -globe version 2.1	84
A.3.1	\mathcal{A} -globe upgraded to the JAVA 2 edition 5.0	84
A.3.2	Library version handling	84
A.3.3	Message Transport layer.....	85
A.3.4	Topic messaging.....	85

A.3.5 Class Finder	85
A.3.6 Directory Service	86
A.3.7 Sniffer Improvements	86
A.3.8 Communication Analyzer	86
A.4 New features and Enhancements in A-globe version 2.3	87
A.5 New features and Enhancements in A-globe version 2.4	88
A.6 Future development possibilities of A-globe	88
References	90

ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER.

Acknowledgement of Support: This material is based upon work supported by the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Laboratory, under contract FA8655-02-1-3069.

Disclaimer: Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Laboratory.

The Contractor, Professor Michal Pechoucek, hereby declares that, to the best of his knowledge and belief, the technical data delivered herewith under Contract No. FA8655-04-1-3044 is complete, accurate, and complies with all requirements of the contract.

April 25, 2006

A handwritten signature in blue ink, reading "Michal Pechoucek". The signature is fluid and cursive, with a long horizontal stroke at the end.

Dr. Michal Pechoucek, head, ATG:

I certify that there were no subject inventions to declare as defined in FAR 52.227-13, during the performance of this contract.

April 25, 2006

A handwritten signature in blue ink, reading "Michal Pechoucek". The signature is fluid and cursive, with a long horizontal stroke at the end.

Dr. Michal Pechoucek, head, ATG:

1

Introduction – Technical Statement of Work

This document provides a final report of the FA8655-04-1-3044 research effort entitled Advanced Agent Methods in Adversarial Environment and funded by AFRL/EOARD. It provides a comprehensive research report detailing the technical results of the project. Therefore, this report incorporates major parts of two previous reports and adds the research results achieved during the last phase of the project.

Present delivery consists of this report and demonstration CD and contains the elements listed hereafter:

1. Paper and electronic version of this report
2. Source code of the current version of **A-globe** multi-agent platform and **ACROSS** scenario including the project-specific mechanisms such as trust model and stand-in agents
3. Demonstrations that replicate the principal experiments presented in this report: trust modelling and stand-in network optimization
4. Demonstration videos presenting the **ACROSS** domain, **A-globe** platform and stand-in optimization

This delivery is going to be delivered in electronic and paper forms. The final delivery will be available for download from:

`ftp://adversarial:FA8655-04-1-3044@agents.felk.cvut.cz`

1.1 Introduction

In the project white-paper the technical work has been specified by means of the four specific research targets that we address in the following:

- **RT1: *Competitive and Adversarial Domains*** Our contributions to the first research target are presented in Chapter 4. ACROSS domain (generic scenario) with self-interested agents featuring private preferences and competitive environment was designed and developed. Upon this scenario, we have added the specific support for adversariality modelling: the bandit agents. And finally, we have leveraged the scenario to simulate the natural disasters and humanitarian effort planning in the environment with competitive and adversarial agents.
- **RT2: *Reasoning in Competitive and Adversarial Domains*** The main contribution to this research target is an innovative lightweight trust model that is suitable for embedded devices. Its crucial properties are: (i) use of efficient yet rich fuzzy arithmetics to represent the uncertainty, coalition cooperation concept that is included into the model, robustness with respect to significant observation noise and autonomous environment adaptation. This model is presented in dedicated Chapter 5.

To further extend the use of the model output, we have designed a mechanism for efficient, trust-bases negotiation in adversarial domains, presented in Section 7.3. This mechanism tightly integrates the trust model with agent’s reasoning (implemented by operations research techniques in our model) and social knowledge.

- **RT3: *Reasoning in Environments with Communication Inaccessibility*** In the scope of this target, we present two principal results: an analysis of existing solution for cooperation in inaccessible environment, including relaying, various middle agents, social knowledge use and stand-in agents designed in course of the previous project. In the comparison, we offer the experiments that analyze the usefulness of each solution in diverse states of environment accessibility as these are defined in the inaccessibility model developed as a part of the deliverable under RT4. During the project, we have discovered that the main obstacle in use of all the above technologies is the efficiency of the solution in the highly dynamic environments (typically ad-hoc networks with mobile nodes). Therefore, we have devised a generic optimization technique for stand-in agents that has several interesting aspects: high system robustness, rapid convergence in both number of agents and messages and last, but not least the mechanism works only with local information and achieves the global optimum reasonably fast. All the findings related to this research target are included in Chapter 6.
- **RT4: *Measuring Properties of the Community*** In the domain of community modelling, we present two contributions defined in Chapter 2 and Chapter 3 respectively: formal model of adversariality and a formal model of accessibility. The first model answers a crucial and non-trivial question, as it tries to draw meaningful boundaries between competitive, self-interested and adversarial behavior. The second model doesn’t define the accessibility, as this definition is a result of the previous effort – it provides a meaningful model of accessibility using the random graph theory and describes important states of accessibility in a multi-agent system. Therefore, it is instrumental for the development of any inaccessibility solution.

Besides these tasks, we have invested in the development of several new features of **A-globe** platform as described in Appendix A. Some of these requirement were directly driven by the needs of our research scenario, while the others (e.g. migration support for libraries) were added to address the demands from the AFRL platform users.

As stated earlier, this document is a final report of the presented research effort. As such it builds on the results presented in the interim reports delivered earlier. Appropriate linkage to the previously published reports is denoted in Table 1.1. This approach makes the report a self-contained document.

Results of the research project were presented at several conferences and other forums. List of publications includes:

Chapter 1 – Introduction – Technical Statement of Work	new in FR
Chapter 2 – Formal Model of Adversarial Behavior	IR1, FR
Chapter 3 – Inaccessible Environment and its Properties	IR1
Chapter 4 – Modelling Inaccessibility and Adversarial Behavior	IR1, IR2, FR
Chapter 5 – Using Trust for Coalition Formation in Adversarial Environment	IR1, IR2
Chapter 6 – Solving Inaccessibility in the Adversarial Environment	IR1, IR2
Chapter 7 – Efficient Teamwork in Inaccessible and Adversarial Environment	new in FR
Chapter 8 – Conclusions and Future Work	new in FR
Appendix A – Progress in A-globe Multi-Agent Platform Development	IR1, IR2, FR

Table 1.1. Linkage of this report (FR) to the previous published reports: Interim Report 1 (IR1) and Interim Report 2 (IR2). For each chapter, we list the reports where it was introduced or updated.

- Šišlák, D. - Rollo, M. - Pěchouček, M.: *A-globe: Agent Platform with Inaccessibility and Mobility Support*, in Cooperative Information Agents VIII, LNAI-3191, Springer-Verlag, Heidelberg, 2004
- Rehák, M. - Pěchouček, M. - Tožička, J. - Šišlák, D.: *Using Stand-In Agents in Partially Accessible Multi-Agent Environment*, Proceedings of Engineering Societies in the Agents World, Springer-Verlag, 2005
- David Šišlák, Michal Pěchouček, Martin Rehák, Jan Tožička, Petr Benda: *Solving Inaccessibility in Multi-Agent Systems by Mobile Middle-Agents*. In: Multiagent and Grid Systems - An International Journal, IOS press (accepted for publication), ISSN 1574-1702, 2005
- David Šišlák, Martin Rehák, Michal Pěchouček, Milan Rollo and Dušan Pavlíček: : **A-globe: Agent Development Platform with Inaccessibility and Mobility Support**. In: Software Agent-Based Applications, Platforms and Development Kits (chapter in a book), p. 21–46, Birkhauser Verlag, Berlin, ISBN 3-7643-7347-4, 2005
- Marco Carvalho, Michal Pěchouček, Niranjan Suri: *A Mobile Agent-Based Middleware for Opportunistic Resource Allocation and Communications*, in Proceedings Workshop of Defence Application of Multi-Agent Systems (DAMAS), AAMAS2005, Utrecht, 2005
- Michal Pěchouček, Martin Rehák, Vladimír Mařík: *Expectations and Deployment of Agent Technology in Manufacturing and Defence: Case Studies* Proceedings of AAMAS 2005, ACM, July 2005
- Martin Rehák, Michal Pěchouček, Petr Benda, Lukáš Foltýn: *Fuzzy Number Approach to Trust in Coalition Environment*, Proceedings of AAMAS 2005, ACM, July 2005
- Martin Rehák, Michal Pěchouček, Petr Benda, Lukáš Foltýn: *Trust in Coalition Environment: Fuzzy Number Approach*, in Proceedings of the Workshop Trust in Agent Societies, AAMAS2005, Utrecht, 2005
- Martin Rehák, Michal Pěchouček, Jan Tožička: *Adversarial Behavior in Multi-Agent Systems*, Proceedings of CEEMAS 2005, Springer-Verlag September 2005
- Martin Rehák, Lukáš Foltýn, Michal Pěchouček, Petr Benda: *Trust Model for Open Ubiquitous Agent Systems*, Proceedings of IEEE/WIC/ACM IAT-2005, IEEE Computer Society Press, September 2005
- David Šišlák, Martin Rehák, Michal Pěchouček, Petr Benda: *Optimizing Agents Operation in Partially Inaccessible and Disruptive Environment*, Proceedings of IEEE/WIC/ACM IAT-2005, IEEE Computer Society Press, September 2005
- Martin Rehák, Michal Pěchouček, Jan Tožička: *Adversarial Behavior in Multi-Agent Systems*, Proceedings of EUMAS 2005, workshop, accepted for publication

Besides, the **A-globe** platform with **ACROSS** scenario has been awarded following awards:

CIA (Cooperative Information Agents) **System Innovation Award** at the CIA workshop in Erfurt, 2004;

The **2005 IEEE/WIC/ACM WI-IAT** Joint Conference: **The Best Demo Award**, in Compiègne, September 2005

2

Formal Model of Adversarial Behavior

The first step towards a realistic modelling of the adversarial environment is a correct definition of adversariality in a multi-agent system. This chapter is dedicated to the definition, that is based on recognized foundations from law, economics and conflict theory, and is consistent with existing work in the computer science domain. This definition is of foremost relevance in open systems, where the collaborating agents belong to more than one party and represent different, even if overlapping interests.

2.1 Introduction

Openness, implemented by ad-hoc integration – both syntactic and semantic interoperability – comes with a price. In such environments, we can no longer assume that the agents are cooperative. The agents in these system can have their own, sometimes partially or even completely antagonistic goals and they often compete for the shared resources or opportunities.

In such environments, we must ensure that the system as a whole will autonomously maintain its sustainability and efficiency, that self-interested agents will be able to agree at least on some goals and that their cooperation will leverage their capabilities. To do so, agent researchers frequently introduce the concepts from microeconomics and game theory, most notably mechanism design [20]. Mechanism design is used to design interaction patterns in the system to promote globally desirable behavior and reduce incentive for undesirable behavior. However, despite the fact that it will provide the basis of the algorithms and protocols of such systems, it still suffers from some serious limitations. Mechanism design techniques have achieved some spectacular results, but their applicability is in general restricted to static environments, where the fine-tuned mechanisms perform well. However, the problems like bounded rationality of the agents, their possible polyvalence, strategic behavior and willingness to keep some of their knowledge private can not be completely addressed by the current mechanisms [26].

Alternatively, similar results can be achieved using norms [19], enforcing flexible social commitments [47], adjustable policies [72] or trust and reputation [16, 60]. But in general, these approaches rely on the fact that the agents are able to distinguish the undesirable behavior in all possible contexts. Therefore, as the system adapts to its environment, the norms, policies and trust mechanisms must be adapted as well to avoid becoming an obstacle of system efficiency, rather than to support it.

In this work, we will look at the problem from somewhat different perspective – after the brief analysis of existing approaches in the multi-agent field, we will use the conflict theory and some fundamental principles from the economy and law to consistently define the adversarial behavior in the multi agent system (Section 2.2) and provide a specific example that instantiates the definition in Section 2.3.

Currently, adversariality in the multi-agent systems is a concept that has been defined in many different contexts. Most of the current definitions are mutually exclusive, but they provide a valuable guidance in our attempt to formalize the definition using their overlaps.

In the field of multi-agent systems, **adversarial planning** was introduced [76] to analyze the behavior of two opponents. However, even if the approach remains interesting due to the analysis of planning in conflicting environment, it is of limited importance for the definition of adversarial behavior. The definition proposed by the authors, where they define adversariality by "opposite goals" doesn't fit our needs, as the agents in the general system we consider (i) are not always adversarial and at least some of their goals are common, (ii) communicate by other means than pure actions, (iii) have asymmetric and partial knowledge and (iv) are deliberative, therefore possibly adversarial within the limited scope of time or issues.

In the mechanism-design field, [26] defines adversarial entities as the entities who's goals can not be described by a utility function and assumes these actors to be irrational. This definition well captures the fact of bounded rationality of agent perceptions - some agents can have goals that are impossible to capture and understand during normal system operations and that are justified by large scale (time or space) behavior of their owners.

We shall use the conclusions from the field of the conflict theory to (i) determine the defining properties of adversariality as they are currently understood.

In his contribution [25], **James Fearon** analyzes the war between two or more perfectly rational states. For Fearon, the most important distinguishing property of the war from the rationalist point of view is the war's **ex-post inefficiency** – he argues that the states can reach the same result by negotiation, eliminating the cost of the adversarial actions: "*...ex-post inefficiency of war opens up an ex-ante bargaining range...*"[25]. This is clearly visible from the simple conflict specification proposed by author.

In the work of **Posner** and **Sykes** [57], approaching the problem of optimal war from the legal perspective, the aggression (unilateral beginning of the war) is defined as an action that is *socially undesirable and imposing net social cost*, while the authors assume that the aggression is motivated by the expected profit of the aggressor, either as a result of war or the threat. They argue that this definition of aggression is consistent with the studies on the economics of crime [9], where the *gains of criminal are smaller than the social cost of act*.

In his breakthrough article, **Gary Becker** [9] analyzes the economics of crime, incentives of criminals, their economic motivation and dissuasive effect of punishments and functional justice system. Besides the definition of criminal activity stated above, the notion of indirect costs is also important. Costs of crime are not only direct, but we must consider the cost of law enforcement as inseparable from the direct crime costs. In a multi-agent system, the well designed mechanisms and trust maintenance models come with a cost that may harm the system efficiency through their computational requirements and other associated requirements. This doesn't mean a refusal of the principle of trust maintenance and mechanism design, but it means that the mechanism must be efficient and well adapted to the current environment.

2.2 Adversarial Behavior Definition

This section is devoted to the formal definition and characterization of adversarial behavior in the multi-agent systems. We will depart from the conflict theory premise stated above that conflict is an *ex-post inefficient* method of resolving competitive issues that imposes a net cost on the society, and we will base our formal definition on these notion. Similar classification was done in [13], but focused on interaction between different types of agents rather than on definition of types of behavior and didn't use the conflict theory. However, some preliminary technical definitions are necessary.

In the following, we will use capitals to denote agents.

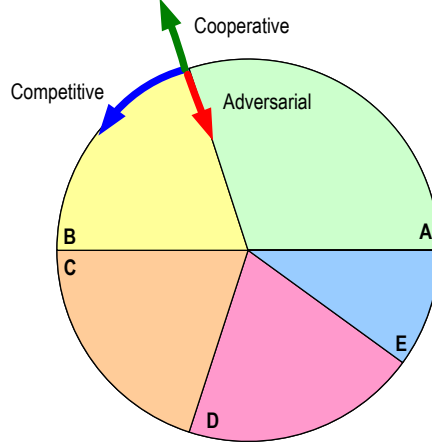


Fig. 2.1. Cooperative, Competitive and Adversarial actions. Pie represents the total utility u and individual utilities u_A, u_B, \dots . We can see that purely cooperative action increases social welfare, purely competitive action doesn't modify the social welfare, but only changes its distribution among agents, while the purely adversarial action reduces the social welfare without any benefit for the agent. In practice, real actions are rarely pure and are a combination of the above types.

Utility is defined as "a value which is associated with a state of the world, and which represents the value that the agent places on that state of the world" by [46].

To simply state our problems, we will define a simple abstract game model featuring agent set $Ag = \{A, B, C, \dots\}$ with the agents playing a non-extensive (single round) game that is not strictly competitive – sum of all agents' utilities is not constant. Each agent X has a set of available actions denoted a_X^* , with actions $a_X^i \in a_X^*$ (whenever possible, we only write a_X). From this set, agent selects its action using its strategy. We suppose that all agents do the selection at one moment and therefore the selected actions are independent. The final state, *outcome* of the game¹ $o(a_A, a_B, \dots)$ is determined by strategies of the agents and determines both the individual agents' utilities $u_A(o), u_B(o), u_C(o), \dots$ and the social choice function $u(o) = u_A(o) + u_B(o) + u_C(o) + \dots$, considered to represent the *social welfare* [20].

In this simplistic game, we can define cooperative, competitive and adversarial behavior in accordance with the principles from mentioned above. As the social outcome is known after the end of the game, the definitions bellow also evaluate agents' actions ex post. Simplified graphical form of the definitions is presented in Fig. 2.1.

In the cooperative environment, all agents do share a single utility function.

Definition 2.1 We say that agent's A action a_A^{coop} is a **cooperative action** if it maximises social welfare:

$$\text{coop}(a_A^{coop}) \Leftrightarrow u(a_A^{coop}, a_B, \dots) = \max_{a_A^i \in a_A^*} u(a_A^i, a_B, \dots)$$

The complete opposite is the self-interested environment, where the agents are trying to maximise their profit.

Definition 2.2 We say that agent's A action a_A^{si} is a **self-interested action** if it maximises agents' individual utility:

¹ The exact form of the outcome is irrelevant, if we are able to obtain the utility values. To simplify the notation, we will also write $u(a_A, a_B, \dots)$ instead of technically more correct $u(o(a_A, a_B, \dots))$.

$$\text{si}(a_A^{si}) \Leftrightarrow u_A(a_A^{si}, a_B, \dots) = \max_{a_A^i \in a_A^*} u_A(a_A^i, a_B, \dots)$$

In many contexts, the terms self-interestedness and competitiveness are considered to be synonymous. However, we consider the competitiveness to be more strict. In [13], self interestedness is defined as not taking the utility of the others into the consideration while maximizing own utility, while [31] requires the trust between competitors, allowing them to avoid globally undesirable outcomes. In the systems with carefully programmed mechanisms, the results are equivalent in both cases. However, in many real-world cases the total utility may decrease, even if each agent optimizes locally (a prison dilemma is a nice example of this situation [5]).

In the competitive environment, agents select actions to maximize their own private utility, but they restrict their choice to the actions that at least conserve the social welfare.

Definition 2.3 *We say that agent's A action a_A^{comp} is a **competitive action** provided that it maximizes individual profit while does not allow drop of the social welfare:*

$$\text{comp}(a_A^{comp}) \Leftrightarrow u_A(a_A^{comp}, a_B, \dots) = \max_{a_A^i \in a_A^*} u_A(a_A^i, a_B, \dots),$$

where $\forall a_A^i \in a_A^{**} : u(a_A^i, a_B, \dots) \geq u(a_B, \dots)$.

The expression $u(a_B, \dots)$ represents hypothetical outcome of the community in the case when the agent A performs no action or is not in the community at all. This situation is illustrated later on Figure 2.2 by \sim symbol.

Finally, let us try to define the concept of the adversarial action. Let us do it in several steps. The most intuitive definition would be that the adversarial action is such an action that is deliberately preferred to another action that is equally achievable but has got higher social welfare utilities.

The main trouble with this definition is that it contains all actions that are not cooperative including self-interested agents who are simply motivated by an increase of their individual utility and they do not consider the information about social welfare in their decision making.

This is why we make an attempt to define an adversarial action as an action that significantly decreases the social welfare while it causes loss or provides only small profit to the actor of the action.

Definition 2.4 *We say that agent's A action a_A^{adv} is an **adversarial action** if: $\text{adv}(a_A^{adv}) \Leftrightarrow \exists a_A^i \in a_A^* :$*

1. $u(a_A^{adv}, a_B, \dots) \ll u(a_A^i, a_B, \dots)$ and
2. $u(a_A^i, a_B, \dots) - u(a_A^{adv}, a_B, \dots) \gg u_A(a_A^{adv}, a_B, \dots) - u_A(a_A^i, a_B, \dots)$.

Similar meaning of the definition can be paraphrased as follows:

Definition 2.5 *We say that agent's A action a_A^{adv} is an **adversarial action** if: $\text{adv}(a_A^{adv}) \Leftrightarrow \exists a_A^i \in a_A^* : u(a_A^{adv}, a_B, \dots) \ll u(a_A^i, a_B, \dots)$ and $u_A(a_A^{adv}, a_B, \dots) \lesssim u_A(a_A^i, a_B, \dots)$.*

The definitions 2.4 and 2.5 above states that the adversarial action a_A^{adv} selected by A from the set a_A^* hurts the social welfare without strong incentive. To make the formalism simpler, we have assumed that there is only single action a_A^{adv} of agent A that hurts the social welfare. There are several interesting points to consider in the general definition.

The first point is the non-emptiness of the set $a_A^* \setminus \{a_A^{adv}\}$ - we don't consider the behaviour with no alternative as adversarial.

Motivation and justification of the adversarial action is closely related to two relational operators used in the definition: \ll and \lesssim . The first inequality \ll signifies that the agent shall not cause

significant harm to the common welfare, while the inequality \lesssim^2 means that the agent remains self-interested and it will not lose a significant part of its welfare to save the utility of other agents. The concept is illustrated by Fig. 2.2. In this context, it is important not to take our simplification of the game formalism literally and to consider only immediate payoff as the utility – in most systems, agents expect to encounter their partners again in the future and we suppose that the attitudes of their partners towards them and expected future profits are included in the utility u_X^3 . Formally, we may pose:

Definition 2.6 We say that **action** a_A^j of agent A is **rationaly adversarial** if it is both self-interested and adversarial. In the action is not self-interested and is adversarial, it is **irrationally adversarial**.

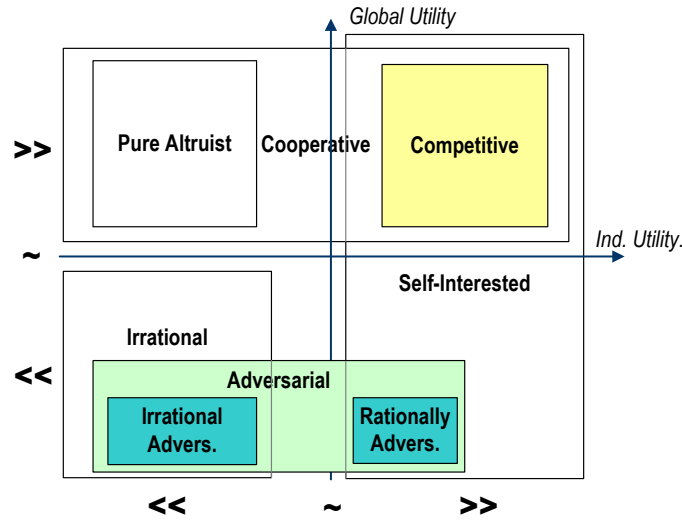


Fig. 2.2. Classification of action with respect to global utility (social welfare) and individual utility of acting agent. The \sim symbol corresponds to the situation where the agent A performs some neutral action or is not in the community at all, i.e. $u(a_B, \dots)$.

2.2.1 Properties of Adversarial Action

Pareto-Optimality

In this context, we may mention the relationship between adversariality and Pareto-Optimality⁴.

An outcome of an adversarial action is not Pareto optimal. Rationally adversarial action is not Pareto optimal in the situations where the agents may negotiate and transfer the utility - in such situations, the agents may always transfer enough utility to motivate the adversarial agent to behave

² We actually mean that the agent has no, or very little motivation to make an adversarial move. In Def. 2.6, we treat the special case when we fall into the \sim case.

³ In this point, we are consistent with the utility definition given above. We have omitted the explicit future gains member in the definitions to simplify the notation by using this broader definition of utility.

⁴ Following [41], we denote as o^* a set of all achievable outcomes and we define: Outcome o is considered to be **Pareto optimal** if : (i) it is achievable (i.e. $o \in o^*$) and (ii) not majored by any other outcome $o' \in o^* \setminus \{o\}$, where we define majoring as: $\forall_{X \in Ag} u_X(o') \geq u_X(o)$ and $\exists_{X \in Ag} u_X(o') > u_X(o)$.

cooperatively, therefore achieving socially acceptable outcome. When the utility is not transferable (e.g. indivisibility as defined in [25]), the set o^* is severely restricted and even an action that causes the overall social loss may be considered non-adversarial due to the lack of alternative. In the irrationally adversarial case, Pareto optimality does not hold neither, as the utility is lost both by adversarial agent and the society as a whole.

As stated above, an outcome of an adversarial action is not Pareto optimal in the situations:

- where the agents may negotiate and transfer the utility, and
- the action is irrationally adversarial – $u_A(a_A^{adv}) < 0$.

This property can be used mainly in the situation where the community consists of a coalition of mutually trusted actors and one agent whose adversariality is subject of investigation.

On the other hand, Pareto optimality as such doesn't preserve social welfare (due to the indivisibility), it only ensures that all agents behave rationally given the knowledge about the action of the others.

Intentionality of Adversarial Action

Another point to address is the predictability of the outcome that is closely related to intentionality of the respective adversarial action. The uncertainty of o arises from the simultaneity of all players' moves, while the uncertainty of values $u_X(o)$ and $u(o)$ exists due to the privacy of functions u_X . This seems to make the definition useless – but social knowledge and norms can provide solutions. In most situations, the individuals are able to estimate the actions of others (denoted a_X^{exp}) and the effects of different outcomes on their utility.

Therefore, without considering norms, we pose:

Definition 2.7 *We say that action a_A^{adv} of agent A is intentionally adversarial if:*

1. *the action is adversarial according to definition 2.5: $\text{adv}(a_A^{adv})$, and*
2. *the agent A **knows that** it is adversarial: $(\text{Bel } A \text{ adv}(a_A^{adv}))$, and*
3. ***expects** the actions a_{A_i} of the other agents to happen next: $(\text{Bel } A \text{ X}(\text{Perform } A_i a_{A_i}))^5$, and*
4. ***implements** the action a_A^{adv} : $\text{X}(\text{Perform } A a_A^{adv})$.*

More specifically, the lack of norms or conventions is a possible cause of unintentional adversariality – the adversarial outcome may arise due to the limited computational power or knowledge of agents, private knowledge or the environmental noise. Important question of attribution must be solved by each agent – we can not expect that all agents will agree on the cause of the common loss.

Existence of shared normative system reduces the uncertainty regarding the expected actions of other agents (a_X^{exp}). In our future work, we will use this system in the adapted definition of adversarial action. On the other hand, definition 2.5 remains valid, as it provides feedback for update of the normative system in changing environment.

Realisation of Adversarial Action

So far, we have defined adversarial action, rational adversarial action and intentional adversarial action. However, we still have to define *adversarial agent*. The way how the concept of adversarial action is mapped into adversariality of an agent is through actual realisation of an adversarial action.

⁵ Where the X operator is a temporal logic operator representing validity of a formula in the next time step or near future in the case of environment with continuous time.

Definition 2.8 We say that *agent A* is **adversarial** (or there exists **adversarial behavior** performed by the agent A) if the agent A performed at least one adversarial action in the past: $adv(A) \Leftrightarrow \exists a_A^{adv} :$

$$adv(a_A^{adv}) \wedge P(\text{Perform } A \ a_A^{adv})$$

In the definition, we assume that the predicate *adv* classifying the actions is defined according to the property 2.5, the P operator to be a temporal logic operator representing validity of a formula in the past and the operator **Perform** linking an agent and the action performed by the agent.

There are clear extensions of this definition of adversarial behavior that define adversariality in a time window, or agent's adversarial behavior with relation to a specific agent community. In the definition 2.8 we assume by default the whole of the community as a target of agent's adversariality and the whole past as the relevant time window.

Implication of Adversarial Behaviour

We are interested in the impact of the adversarial action on the global social welfare of the community *Ag*. We say that:

- decrease of social welfare *does not imply* existence of an adversarial behavior in the community,
- unnecessary decrease of social welfare *implies* existence of an adversarial behavior (intentional or unintentional) in the community, while
- existence of an adversarial behavior in the community *does not imply* decrease of social welfare.

For the proof of these statements, let us consider only types of actions according to the definitions 2.1, 2.3 and 2.5. No combination of cooperative and competitive actions may cause an overall decrease of the social welfare, thus an existence of at least one adversarial action is inevitable. In contrary, for a combination of adversarial actions there may exist a compensating combination of cooperative or competitive actions that can be carried out by any member of the community in the finite time *t* so that in *t* the social welfare does not decrease.

The definition 2.8 does not classify performance of an action that has got a direct inevitability (or possibly an option) of an adversarial action as its effect as adversarial behavior.

2.3 Examples of Adversarial Behaviour

2.3.1 Adversariality in Coalition Formation

In this example, we will illustrate rather abstract definitions provided above with the real example, the coalition formation, approaching the problem from the utility side. We will start by introducing the necessary notation. In this section, we consider the coalition to be short-lived and therefore the terms adversarial action of agent *A* and adversarial agent *A* will be used interchangeably.

Using the concept of the marginal utility⁶, we may now define cooperative and competitive behavior in our example.

⁶ Agent's *A* *marginal utility* (*mu*) from joining the coalition *C* (an activity denoted as $A \mapsto C$) is a difference between the agent's utility before and after it joins the coalition ($mu_{A \mapsto C}(A) = u_{A \in C}(A) - u_{A \notin C}(A)$), where $u_{A \in C}(A)$ is a utility the agent *A* (in parentheses) receives as a member of the coalition *C* (situation is described by subscript), while $u_{A \notin C}(A)$ denotes the utility agent *A* receives if it doesn't join the coalition *C*). The marginal utility of a coalition *C* in agent's *A* joining the coalition is defined as a derivation of the collective utility (such as social welfare) of the coalition before and after the agent joins the coalition ($mu_{A \mapsto C}(C) = u(C \cup A) - u(C)$).

We say that agent A is **collaborative** provided that: if an agent A makes an attempt to join the coalition C then always $mu_{A \rightarrow C}(C) > 0$. We shall note that even if all agents are collaborative, the optimum result is not guaranteed. A typical case can be described as follows: $mu_{B \rightarrow C}(C \cup B) > mu_{A \rightarrow C}(C \cup A) \geq 0$ and $mu_{B \rightarrow (C \cup A)}(B) < 0$. If A joins the coalition first, it blocks the entry of B and only local optimum is reached.

We say that agent A is **competitive** provided that: if an agent A makes an attempt to join the coalition C then always $mu_{A \rightarrow C}(A) > 0$ and $mu_{A \rightarrow C}(C) \geq 0$ ⁷. Similarly, we say that agent A is **self-interested** provided that: if an agent A makes an attempt to join the coalition C then always $mu_{A \rightarrow C}(A) > 0$.

As we have already stated before, self-interested agent considers only its own profit while it takes coalition entry decision. Competitive agent is both self interested and collaborative, as it maximizes its own profit, but it at least maintains the social welfare that is represented by the coalition utility. Therefore, in both competitive and cooperative behavior, the social welfare is maintained. This is not necessarily true in the self-interested or adversarial behavior.

In this example, we will use the marginal utility defined above to define adversarial behavior. We say that an agent is **adversarial** provided:

- $mu_{A \rightarrow C}(A) \lesssim 0$
- $mu_{A \rightarrow C}(C) \ll 0$
- agent A makes an attempt to join the coalition C

Informally, an agent is adversarial with respect to coalition C provided that the increase of his direct marginal utility is significantly smaller than the harm (decrease of the total payoff) caused to the coalition.

If the condition $mu_{A \rightarrow C}(A) \geq 0$ holds, agent's action is rationally adversarial, otherwise it is irrationally adversarial, as defined in definition 2.6.

Main advantage of the above definition is that it provides a basis for the detection of adversarial agents, by defining the metrics measuring the adversariality.

Gathering and maintaining such experience is not trivial. However, we may reuse the existing work on trust, where one of the components of the trust [16] – intentional trust (willingness) – is a complement of intra-community adversariality defined above. Therefore, if we establish a reasonable value for trust (that may be actually lower, due to the capability trust), we may deduce an acceptable estimation of agent's adversariality. This is one of the considerations taken into account for the development of trust model described in Chapter 5.

2.3.2 Adversariality Flight-plan Deconfliction

The other motivation example we wanted to illustrate the idea of adversarial action is from the less abstract domain of UAV (Unmanned Aerial Vehicle) flight-plan deconfliction. Let us operate two UAVs A and B . We have situation where the two UAVs are facing a collision and they individually deliberate about the actions $d(A)$ – the UAV A making the deconfliction manoeuvre, the action $d(B)$ – the UAV B making the deconfliction manoeuvre and $d(A, B)$ – the both UAVs making the deconfliction manoeuvre. As the UAVs are different, they loss of their individual utility associated with the manoeuvre is different. Let us assume:

- $mu_{d(A)}(A) > mu_{d(B)}(B) > mu_{d(A, B)}(B) + mu_{d(A, B)}(A)$
- $mu_{d(A, B)}(A) > mu_{d(A)}(A)$,
- $mu_{d(A, B)}(B) > mu_{d(B)}(B)$

⁷ Note that 0 represents here $u(a_A^B, \dots)$ expression from the definition 2.3 as the marginal utility doesn't change without any action of agent A

The A UAV is smaller than the B UAV, which implies that it is cheaper for A to make the manoeuvre than it is for B . However B making the manoeuvre is even cheaper than sum of costs for both UAVs making the manoeuvre. However, for each UAV it is cheaper to participate in a collective manoeuvre than doing the manoeuvre individually.

In the *cooperative environment* the UAVs with conflicting plans do minimize overall disruption and fuel consumption while solving deconfliction problems⁸:

$$- A, B : d(A) \succ d(B) \succ d(A, B) \succ \neg d(A, B)$$

Both the cooperative UAVs have the same strategy, that is the smaller UAV deconfliction is preferred to the bigger UAV deconfliction that is preferred to both plans deconfliction and which is preferred to a confliction.

In the *competitive environment* each UAV minimizes its own plan disruption and fuel consumption, but conserves overall welfare:

$$\begin{aligned} - A : d(B) \succ d(A) \succ d(A, B) \succ \neg d(A, B) \\ - B : d(A) \succ d(B) \succ d(A, B) \succ \neg d(A, B) \end{aligned}$$

More specifically, if two UAVs would collide and only one evasive maneuver is necessary, it will try to make the other UAV divert from its course, but without compromising the security.

In the *self-interested environment* each UAV minimizes its own plan disruption and fuel consumption, regardless of the others.

$$\begin{aligned} - A : d(B) \succ d(A, B) \succ d(A) \succ \neg d(A, B) \\ - B : d(A) \succ d(A, B) \succ d(B) \succ \neg d(A, B) \end{aligned}$$

In case of conflict, it only diverts from its course to protect its own safety.

In the adversarial environment the adversarial UAV causes a significant disruption of the other's plans, or even endangers them. Let us assume $mu_{d(A)}(A) \ll mu_{d(B)}(B)$ (e.g. A is pushing B away).

$$\begin{aligned} - A : d(B) \succ \neg d(A, B) \succ \{d(A, B), d(A)\} \\ - B : d(A) \succ d(A, B) \succ d(B) \succ \neg d(A, B) \end{aligned}$$

If the B flying over an enemy area does not want to be pushed, the situation is as follows:

$$\begin{aligned} - A : d(B) \succ \neg d(A, B) \succ \{d(A, B), d(A)\} \\ - B : d(A) \succ d(A, B) \succ \neg d(A, B) \succ d(B) \end{aligned}$$

2.4 Conclusions

The problem of adversariality in the multi-agent systems is real. While the irrationally adversarial agents may be easy to identify, it may be much more difficult to identify the rationally adversarial behavior, especially if all the agents in the system are self-interested. In this context, the question of *bounded rationality* of agent's reasoning is crucial. For example, some agents may be willing to leave the local optimum to bring the system into the globally optimal (or simply better) state. However, if the other agents in the system lack this insight, they may consider this behavior as adversarial because they fail to see the long-term benefits.

To better illustrate the concept, we will list several accepted causes for the emergence of the conflict between the rational actors. It is easy to realize that most of these causes can plausibly exist in the multi-agent system and shall be considered while designing autonomous agents:

This can happen due to the fact that private information of each agent is not available to the others. This provides one of the causes of miscalculation about capabilities or attitudes of the other

⁸ The symbol \succ stands for collective choice preference

party. Such miscalculation may cause an adversarial behavior, as the agents will not be able to correctly estimate the utility function of the partners. Similarly the agents are often willing to misrepresent the reality about themselves, in order to obtain better payoff or negotiation position in the future. However, if such behavior becomes widespread in the system (It can be often prevented by careful mechanism design.), agents are unable to communicate efficiently. In more sophisticated extension of this behavior, agents can behave strategically and harm the others to gain higher relative power in the long term. In some situations, the system may even become purely competitive – agents or their groups have nothing to gain from cooperation, for example when the payoff is indivisible.

The definition of the adversarial behavior that we present provides a useful complement of the current approaches to the open systems engineering. Even if the system is based on carefully designed mechanisms and/or norms, the changing system social structure and the environment or agent's strategic behavior may modify the system and make it inefficient or dysfunctional. To counter such danger, the agents in the system shall continuously monitor the behavior of the others and their own and detect potentially adversarial actions. In the context of our work, we have implemented a trust model described in Chapter 5 that integrates seamlessly with agent reasoning, is robust with respect to significant environment noise and helps the agents to avoid the collaboration with untrustworthy partners.

Apart from using the model of adversarial behavior for construction of the trust models of the individual agents about the other agents, we see another level of the exploitation potential on the level of agent infrastructure. Multi-agent systems integrations the meta-level components for policy management and adjustable autonomy [64] can make use of the adversariality models for dynamically changing the policies but also for creating norms – this is a valid option for future research, as such approach promises higher efficiency and can be a good fit for high-performance applications like networking.

3

Inaccessible Environment and its Properties

3.1 Measuring Inaccessibility

An important problem is how to quantify inaccessibility in multi-agent systems. In the following we discuss several metrics of inaccessibility that we have been using throughout our research project¹.

Let us introduce a **measure of inaccessibility**, a quantity denoted as $\bar{\vartheta} \in [0; 1]$. This measure is supposed to be dual to the **measure of accessibility** – $\vartheta \in [0; 1]$, where $\vartheta + \bar{\vartheta} = 1$. We will want ϑ to be 1 in order to denote complete accessibility and ϑ to be 0 in order to denote complete inaccessibility. This measure is consistent with the circuit reliability defined in [30]. In the following text we will mostly describe the agents' accessibility while the inaccessibility is its complement.

3.1.1 Agent-to-Agent Accessibility

In this section we will use the random graph theory [11] in order to describe some general properties of inaccessibility in multi-agent systems. Random graph theory has been recently successfully used for theoretical studies of complex networks [2].

In this context, we represent the multi-agent community as a graph. The agents are represented by nodes and available communication links – connections where the information exchange is possible – by edges. Unlike in the general case of agents inaccessibility, the random graphs theory works with an assumption that all edges are present with the same probability p . In our domain, this probability is represented by **link accessibility**: $p = \vartheta$.

The ϑ link accessibility can be determined in two ways. Firstly as ϑ_t :

$$\vartheta_t = \frac{t_{\text{acc}}}{t_{\text{inacc}} + t_{\text{acc}}}, \quad (3.1)$$

where t_{acc} denotes the amount of time when communication is possible while t_{inacc} denotes time when agents are disconnected.

Similarly, we may measure accessibility as a function of communication requests (ϑ_m):

$$\vartheta_m = \frac{|m| - |m_{\text{fail}}|}{|m|}, \quad (3.2)$$

where $|m|$ denotes the total number of messages sent and $|m_{\text{fail}}|$ the number of messages that failed to be delivered. In the following we will discuss ϑ_t while most conclusions apply equally to ϑ_m . The accessibility measure ϑ_t is symmetrical between entities A and B

$$\vartheta_t(A, B) = \vartheta_t(B, A), \quad (3.3)$$

¹ Definition of accessibility was already part of the deliverable [49].

while the accessibility measure ϑ_m is not necessarily symmetrical.

In our model, ϑ_t accessibility depends on the environment agent positions only, while ϑ^m accessibility depends also on other factors, like communication link load or social knowledge availability of the agents.

We have counted the probability of existence of the path between two agents - **path accessibility** - depending on link accessibility in simple mathematical simulation. The result is shown in figure 3.1. Classical result of the random graph theory is that there exists a critical probability at which large

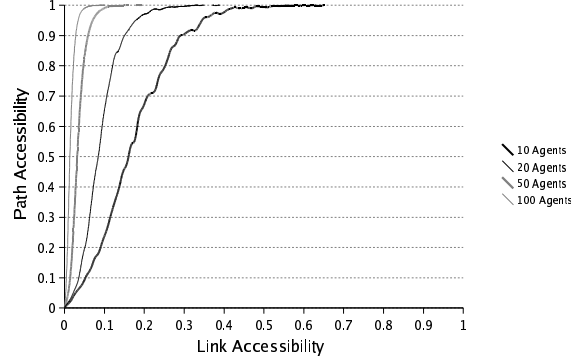


Fig. 3.1. The dependency of path accessibility (probability of existence path between two agents) on link accessibility. This graph is the same for the link accessibility with or without the symmetry.

cluster appears. In our domain, we assume that there is a **critical accessibility** – ϑ^c such that below ϑ^c the agent community is composed of several isolated groups but above ϑ^c most of agents become mutually path-accessible (using relay agents). The ϑ^c value is represented by the quick growth in the figure 3.1. This observation is similar to a percolation transition known in the field of mathematics and statistical mechanics [68]. In the field of multi-agent systems, it means that the relay agents are more efficient than isolated stand-in agents for link accessibility bigger than ϑ^c . Our testing scenario, presented in section 6.2.3 and implemented using actual multi-agent system based on **A-globe** [66] allows to set up and verify properties of both cases.

3.1.2 Accessibility Characteristics

Second relevant result of random graph theory is the average length l^* of a path between any two vertices and the diameter l^d of a graph (i.e. maximal distance between any two nodes). It holds [2]:

$$l^* \sim l^d = \frac{\ln(n)}{\ln(\vartheta_t n)},$$

where n is number of agents.

In our domain, the length of the path says how many relays must be used in order to convey a message between the agents A and B .

Below we summarize several important results:

$\vartheta_t n < 1 \Rightarrow$ the network is typically composed of isolated trees. The diameter is equal to the diameter of trees. (3.4)

$\vartheta_t n > 1 \Rightarrow$ a large cluster is formed. The diameter is equal to the largest cluster diameter and if $\vartheta_t n > 3.5$ it is proportional to $\frac{\ln(n)}{\ln(\vartheta_t n)}$. (3.5)

$\vartheta_t n > \ln(n) \Rightarrow$ the graph is probably totally connected and the diameter is very close to $\frac{\ln(n)}{\ln(\vartheta_t n)}$. (3.6)

These properties are well observable also in our domain (see section 3.2.2).

3.1.3 Accessibility Temporal Characteristics

Agent cooperation, as discussed for example in [34] typically requires that all cooperating agents know about the properties and existence of each other (awareness) and that they are able to communicate (explicitly or implicitly) in order to coordinate their actions. In the distributed and potentially inaccessible coordination cases, we call these properties *remote awareness* and *remote presence*.

When an agent builds **remote awareness**, it tries to let the remote agents include knowledge of its existence and all relevant information into their social knowledge and to let them operate using this information. This process may be implemented using pull or push information retrieval operations. Typical examples are acquaintance models described in section 6.1.3, matchmaking middle agents (6.1.2) or synchronization and search in peer-to-peer networks [24].

When an agent builds a **remote presence**, it does so to operate **actively** in the remote location. As a collateral effect of this action, agent may also build a remote awareness - as in the stand-in case, but this does not necessarily apply when we use middle agents. Examples of this approach are relaying (6.1.1), stand-ins (6.1.4) or broker middle agents (6.1.2).

In a perfectly accessible environment, communication between cooperating agents is unrestricted and therefore the exchange of knowledge, as well as negotiations and commitments are unlimited, making remote awareness and presence perfect. With increasing communication inaccessibility, cooperation quality decreases and the only way to maintain it is to use special techniques for remote awareness and remote presence maintenance.

Therefore, besides the simple accessibility ratio defined above, we shall also analyze relevant temporal accessibility properties. These properties are crucial especially for establishment of remote presence, as most interactions between agents require a sequence of messages to be exchanged following a predefined protocol. If such sequence is broken and some messages are lost, the mutual knowledge about the commitments can not be guaranteed and cooperation may fail. As the loss of communication is not always perceived by both partners, the non-fulfilment of an expected action can harm the reputation of executing parties, with all possibly damaging consequences. This is why the **time of accessibility** - length of the time interval, during which are the entities accessible, can seriously influence the system performance.

To develop our theory, we use a formalism analogous to well known MTBF relations from reliability theory [44] and therefore the mathematical apparatus developed for this theory applies in our case also.

Formally, we may describe the accessibility between two entities by two values - **mean time of accessibility**, denoted τ_ϑ and **mean time of inaccessibility**, denoted $\tau_{\bar{\vartheta}}$. The relation between the accessibility defined above and these values is an obvious one:

$$\vartheta_t(A, B) = \frac{\tau_\vartheta(A, B)}{\tau_{\bar{\vartheta}}(A, B) + \tau_\vartheta(A, B)} \quad (3.7)$$

In the following section, we are going to describe the existing solutions for inaccessibility problem and analyze their use in domains characterized by the defined metrics. The formalism presented in this paragraph will be revisited in paragraph 6.1.5.

3.2 Measure of Accessibility - Experiments

In this section, we will describe a set of accessibility experiments with a multi-agent simulation. The goal of the experiment was to validate the relevance of the theory presented in the first part (section 3.1) of this chapter on a real multi-agent system. In the same set of experiments, we have also established the boundaries of applicability of various solutions for inaccessibility, as discussed in Chapter 6.

3.2.1 Testing Scenario

For our measurements, we have generated a graph with 10 static nodes placed in 2D space and 7 nodes moving among the static nodes. The link accessibility represents a limited-range radio communication.

The graphs were generated from 11 sets of measurements, each with different communication range that defines the inaccessibility within the system.

3.2.2 Results

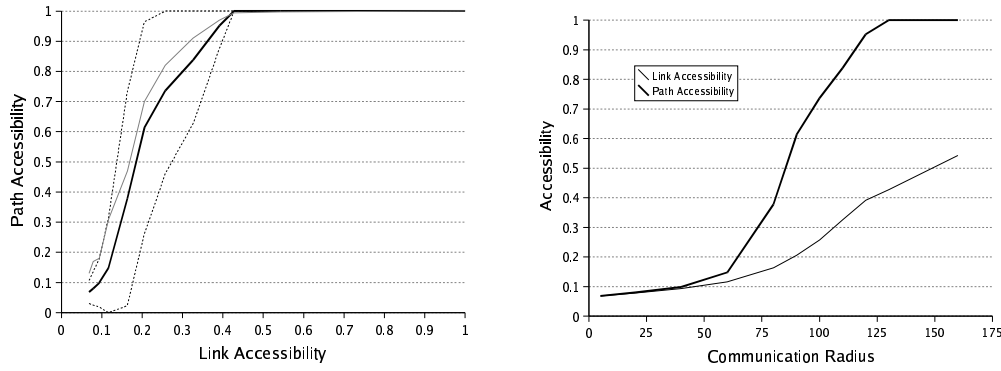


Fig. 3.2. *Left:* The dependency of path accessibility on link accessibility in our test scenario. The dot lines show average deviation of values. The gray thin line shows theoretical value for random graph with 10 nodes (see fig. 3.1). *Right:* The dependency of link and path accessibility on communication radius.

On figure 3.2, we can identify three major states of the community from the accessibility point of view, as defined in section 3.1.2. At first, before the communication radius reaches 60, static community members are isolated and information is not transmitted (see Eqn. 3.4), but only carried by moving entities. In this state, path accessibility is not significantly different from link accessibility. Therefore, probability of relaying is almost negligible.

Then, with increasing communication radius, larger connected components do start to appear, covering several static and mobile entities and allowing the use of relaying over these portions of the graph. This phase appears around the percolation threshold, that can be observed above radius of 80, corresponding with link accessibility of 0.2. This state is characterized by important variability of connected components. Due to the dynamic nature of our system, these components are relatively short-lived, resulting in a high variability of the system, as we can see on figure 3.3. Path accessibility in the community may be described by relation 3.5

In the last state, when communication radius is above 120 and link accessibility reaches 0.4, the entities become almost completely connected. We say that accessibility is **achievable**, as defined

above. This state of the community is described by relation 3.6. System properties does not change when we further increase the radius and link accessibility.

The dynamic nature of our network near percolation threshold is clearly visible on the following graph (figure 3.3), where we present the number of locations visible from one randomly chosen entity of each location type over a period of time. As we are near the percolation threshold, in the state described by Equation 3.5, we can observe the appearance of relatively large, but short lived, accessible components.

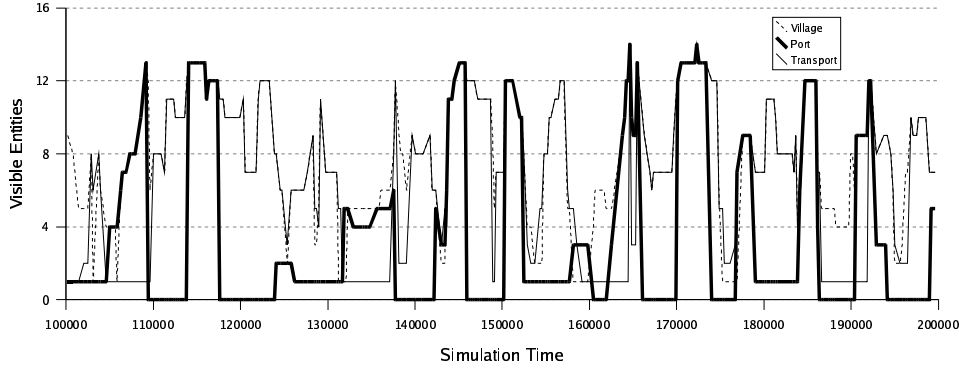


Fig. 3.3. Number of visible entities for different types of entities in our scenario, for communication radius of 80, near percolation threshold.

In the figure 3.3, we may also note that the transport is accessible from significantly more locations than static elements. As this holds for all transports in our scenario, a parallel with scale-free networks [8] arises. In these networks, a small number of nodes called hubs has significantly more connections with others than the rest, while in the random networks most nodes have the same number of adjacent edges. In this respect, transport platforms with stand-ins on them serve as hubs of our system, spreading the information as they roam through the map. This hypothesis is validated by the subsequent experiments on the identical domain, presented in Chapter 6. This chapter also introduces and evaluates the concepts that help us to manage and solve the inaccessibility.

To summarize the findings, the experiments as they are presented prove that the random-graph theory with other elements from mathematical network theory is a correct formalism for inaccessibility modelling in mobile distributed agent systems.

4

Modelling Inaccessibility and Adversarial Behavior

In order to experiment with and demonstrate the investigated research concepts, we need an environment that supports modelling communication inaccessibility and modelling adversarial behavior. We have designed and developed the generic ACROSS scenario deployed within the *A-globe* platform. This generic scenario is agent-based and very flexible – we can augment the baseline version (Section 4.1) to support inaccessibility (Section 4.2), adversarial behavior (Section 4.3) or even to simulate disasters that can be solved by cooperating humanitarian agents, as described in Section 4.4.

During the course of the project, the scenario development objectives were frequently re-aligned with the needs of the research. On the other hand, the agents in the scenario were being enriched by the new capabilities stemming from the research results, and their final version is a core of the prototype presented in Section 4.4. This double influence has enabled us to orient the research towards plausible, real-world-like problems and to propose efficient solutions to these problems.

4.1 Domain Description – ACROSS

We have designed a specific scenario simulating logistics humanitarian aid provisioning. In our scenario, figure 4.1, we solve a logistics problem in a non-collaborative environment with self-interested agents. Agents that are part of the scenario have no common goals and their cooperation is typically financially motivated.

We have three types of information about each agent [51]. *Public information* is available to all agents in the system. It includes the agent identity, services proposed to other agents and other relevant characteristics it wishes to reveal. *Semi-private information* is the information which the agent agrees to share with selected partners in order to streamline their cooperation. In our case, resource capacity cumulated by resource type is shared within transporters’ alliances (see below). *Private information* is available only to agent itself. It contains detailed information about agent’s plans, intentions and resources.

Following types of agents participate in the scenario as actors:

Location Agents: Location agents represent population and natural resources, figure 4.2 (a). They create, transform or consume resources. As most location agents are unable to completely cover the local demand, they acquire the surplus goods from other locations through one round, sealed bid auctions organized by buyers according to the FIPA CNP protocol [27]. As most Location agents are physically remote, it is necessary to transport the acquired goods from the provider to the buyer. In order to do so, location agents contract ad-hoc coalitions of transporter agents to carry the cargo, figure 4.3.

Transporter Agents: Transporter Agents are the principal agents in our scenario. They use their resources - vehicles, driven by *Driver agents* - to transport the cargo as requested by location

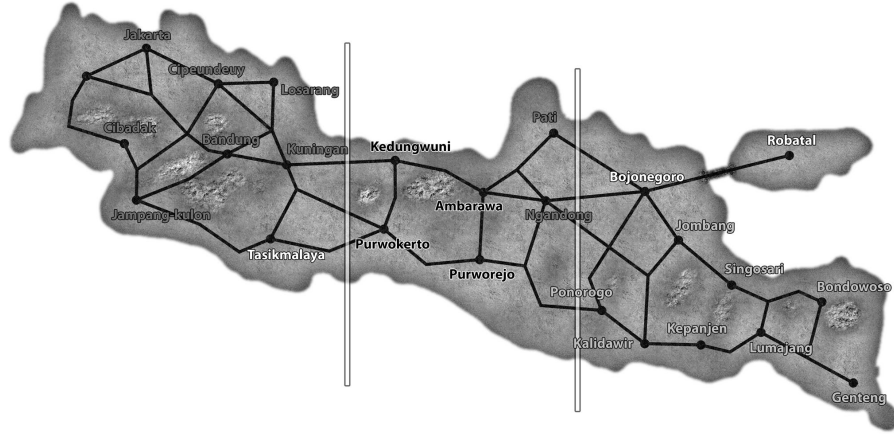


Fig. 4.1. ACROSS scenario. The geography of the island is modelled after the real Java island in Indonesia, with necessary simplifications.

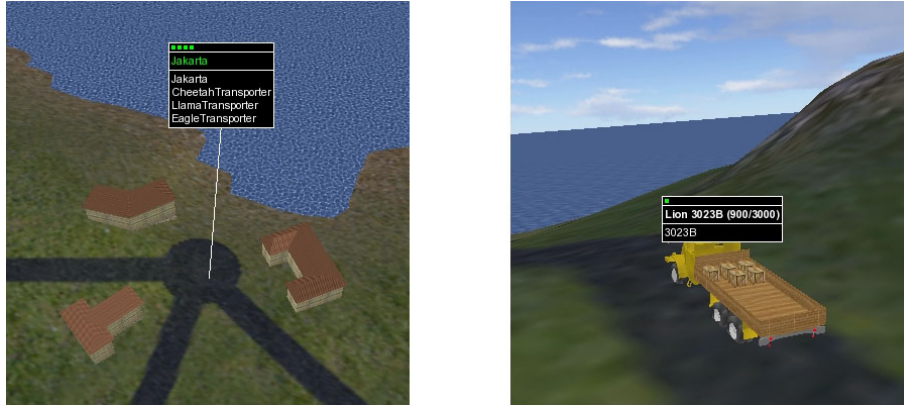


Fig. 4.2. (a) – Location and 3 Transporter agents in a village container; (b) – a Driver agent in a car container

agents. As a normal request exceeds the size that may be handled by a single transporter, transporters must form one-time coalitions in order to increase the coverage and thus to be chosen in the auctions. All transporter agents are self-interested and they don't wish to cooperate with all other transporters. They only pick the partners that are compatible with their private preferences. The compatibility is checked using the public information available about the potential partner and agents' private preferences.

While answering the calls for proposals, the agents must form the coalitions relatively fast and efficiently to submit their bid before timeout elapses. Therefore, they use the concept of alliances, discussed in [51], to make the process more efficient. Alliances are groups of agents who agree to exchange the semi-private information about their resources in order to allow efficient pre-planning before starting the coalition negotiation itself. Using the preplanning, negotiation can directly concentrate on optimization issues, rather than starting from resource query, saving valuable time and messages.

Driver Agents: Driver Agents drive the vehicles owned by Transporter agents, figure 4.2 (b). They handle path planning, loading, unloading and other driver duties.

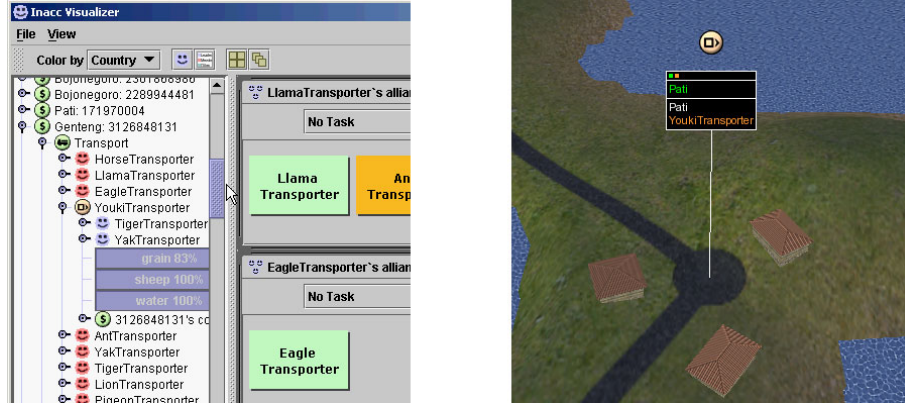


Fig. 4.3. Location agents contract ad-hoc coalitions of transporter agents to carry the cargo

Numbers of agents actually used can vary from project to project, but the basic configuration uses 25 Location agents, each of them in separate container, 25 Transporter Agents distributed among Location agents' containers and 65 Driver Agents, each with its own container. Besides these "active" agents, we do need several services per container to implement platform functions like GIS, directory or migration management. With latest optimizations, this configuration runs on a single PC, greatly facilitating the experiments.

A-globe has been used also for experiments within the underwater surveillance domain. Here a collective coordination and planning of a group of autonomous robots have been investigated. Deployment of **A-globe** in this scenario has been supported by N00014-03-1-0292 project funded by ONR. As this exercise has been of purely collaborative nature, we can hardly use it as background for addressability modelling

4.2 Modelling Communication Inaccessibility

Besides the agents mentioned above, several other agents are used for world simulation purposes, as described in A.2. ACROSS scenario is managed by the following agents:

NodePod Agent simulates the positions and movements of all agent containers (see A.1) in the simulated world. ACROSS world containers are positioned in the graph. Location agents are placed in a selected node, while the vehicle containers move through the graph following the edges - roads. For each moveable container, at least one agent in this container must be able to communicate the decisions about future directions to the NodePod agent and to handle events generated by the NodePod upon arrival to the graph node. NodePod doesn't take any part in road planning or decision making - it plainly simulates the movements of agent container support on the map following the orders from the Driver agents.

For large scale scenarios, we prefer to handle the movements of agents in a central simulation element, rather than in the container itself. This approach, even if slightly less flexible while adding new agents, pays off thanks to the important savings in the number of messages necessary to run the simulation. In most cases, we require the movements to be smooth, requiring at least 10 simulation steps per second. If the movements are managed in a distributed manner, the system would require 600 messages per second just to report the positions of containers. Besides the sheer number of messages, we must take into account the fact that many simulation agents require the knowledge of all agent's position in order to generate their output (for example accessibility). Synchronization then becomes an important issue.

Besides the communication with driver agents, the NodePod agent also provides the updated positions of all containers to all other simulation agents in the master container, especially to the Visibility Agent.

Accessibility Agent is an ES agent that simulates the accessibility between the agent containers. It uses the position data received from NodePod ES agent to determine the distance, updates the data with stochastic link failures specified by the configuration parameter and sends the updates to the containers whose accessibility has changed.

We shall note that the two types of inaccessibility - distance based or caused by the link failures - have very different effects on the processes in the community. In the first case, agents who are inaccessible cannot start any direct interaction and this translates into the suboptimal performance of the system, according to the standard economic theories. On the other hand, if the inaccessibility is stochastic, the interactions can indeed start, but the actors must be aware of the possibility that the link can be broken at any time. Therefore, the agents must adopt an appropriate method for inaccessibility resolution, such as use of stand-ins (see [61]), social knowledge or adopted interaction protocols.

Weather Agent maintains the model of the weather in the various parts of the environment. The weather is generated once per each simulation day and submitted to all Location containers. It is then used to adjust the production or consumption of various resources.

Two additional modules are currently integrated with *NodePod* agent. The **3D visualizer** module ensures the selection and formatting of the data for the external visualizers. Besides the pure position data, this module receives the status messages from agents and displays them in the appropriate visualizers. Due to the important data flow between this module and external visualizers, we were forced to implement an efficient binary protocol for message sending. The **time module** controls the speed at which the simulation runs. It maps the real time to physical simulation step, therefore influencing the basic pace at which the system runs. Besides this fundamental parameter, we can modify the second parameter, that maps the simulation step to simulation day, used to trigger the recurrent agents' actions, such as production or commercial exchanges.

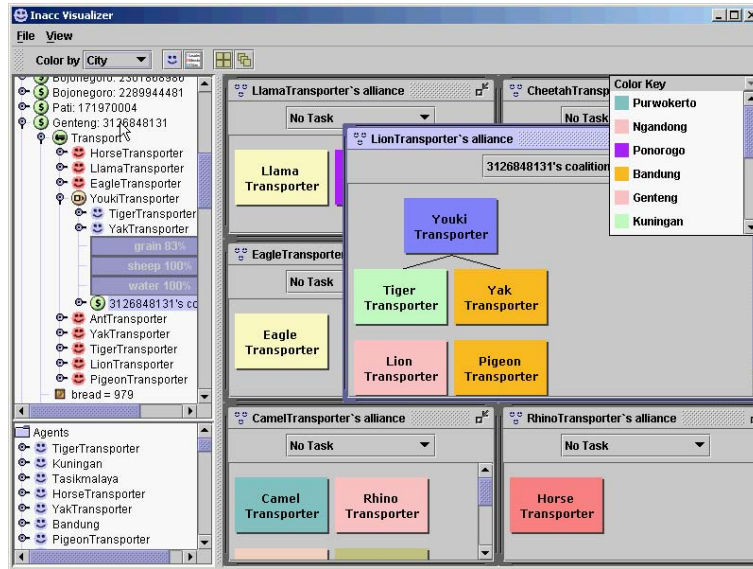


Fig. 4.4. The commercial visualizer GUI

Commercial Visualizer agent visualizes the auctions, including all bids and selected winners, together with the coalitions of transporters that handle the transportation, as shown in figure 4.4.

It also presents the alliances and their formation described above. In contrast to Sniffer or Communication analyzer agents, this agent is scenario dependent. This makes its integration with other scenarios non-trivial, but the specificity makes the presentation efficient and understandable.

In order to obtain the adversarial behavior, the core of the system described above must be enriched with specific components and behavior. These modifications are specific for this project only.

4.3 Modelling Adversarial Actions

In this section, we will present the enhancements of the above generic scenario that are used to implement defection and adversarial actions in the ACROSS . We will also discuss the details of the mechanisms used for coalition evaluation and payoff distribution and emphasize their crucial aspects. The system as described by the previous and this section was used to obtain the results presented in section 6.2.3, where we evaluate the trust model presented in Chapter 5.



Fig. 4.5. The bandit agent action as shown in the 3D visio

One significant change compared to nominal across scenario is an introduction of **bandit** agents, a special type of ES agents (see A.2), who simulate the attacks on the vehicles during their movements between different locations and rob them of their cargo or part of thereof, see figure 4.5. Under normal circumstances, bandits attack stochastically with probability $p_n(LocalHappiness)$ and they are rarely successful. Likelihood of natural bandit attack in a given region is non-decreasing with the increasing suffering of the population - we suppose that poverty increases crime rate. This mechanism is one of the driving forces of the adversariality evolution presented in the next section.

Besides nominal **random attacks** with a relatively low probability p_n , the bandit attacks can be more successful in case of **defection** of one or more agents in the coalition that transports the cargo. Uniform probability $p_d(LocalHappiness, \#ofdefectors)$ is non-decreasing with local happiness and increasing with number of defectors in the coalition. Typically, a relation $p_d \gg p_n$ holds, with the ratios fixed by experiment configuration. Besides the side payment from the bandits, a *single* defector in a coalition obtains an immunity for its cars. If more than one agent defects, no one has an immunity. Technically, defection means informing the bandit about the coalition goals, participating agent and plan - data that is accessible to all members.

When the cargo is lost during the transport, the payments to all coalition members are reduced, as the coalition is paid only for the completed deliveries. The losses are shared between the members.

Transporters are therefore motivated not to cooperate with defectors or frauds and they use the above described reasoning to select their coalition partners.

The problem we solve is analogous to iterated prisoners dilemma (with more than two players) with significant background noise, produced by the following phenomena: (i) The attack probability depends on the length of the transport path and the speed of the car - this fact was intentionally omitted in the trust reasoning experiments (see 6.2.3). (ii) The success of the cooperation (see 5.4.1) is determined by the ratio of the delivered cargo - and each vehicle has different capacity and load. Therefore, a single attack on a fully loaded car can have greater impact than many attacks on the cars with small lots. (iii) The drivers working for the defector are protected. In the very special case when they carry a bulk of the coalitions' cargo, defection may actually increase the success of the coalition.

In the last scenario version, trustfulness values are accessible in the 3D visualizer as soon as enough data is available to evaluating agent. This information is accessible by clicking the Transporter Agent name in the visualizer. Then, the color-coded list with the names of other transporters and defuzzified trust values appears and when the cursor is dragged over the column heading or individual partner, a line in matching color appears between the evaluator and evaluated agent.

4.4 Scenario Evolution: Humanitarian Relief Operation in Adversarial Environment

This section describes the extensions that enhance the existing ACROSS scenario, as described in the previous sections. To study complex coordination and cooperation in such environment, we disrupt the equilibrium of economy on the simulated island by introducing a catastrophe that inhibits the creation of resources necessary for population survival in one part of the island. In this evolution of the domain, existing agents are modified to fit following scenario and new types of agents are introduced (besides bandit agent that is also used for the trust experiments in Chapter 5). After the catastrophe, *Humanitarian agents* described in the suite are introduced to the disrupted environment to fill the gap and to provide the missing resources.

As a result of the catastrophe introduced above and managed by dedicated ES agent, one part of the island will severely lack some of the necessary goods, for example water and grain in case of drought.

4.4.1 Inaccessibility and Communication Failures

Intuitively, we assume that the catastrophe affects the communication infrastructure in the disaster-affected area - see Figure 4.6. Therefore, the Humanitarian agents will deploy their stand-ins as described above to obtain the information about the needs of the population. Moreover, stand-ins will be used also to coordinate the actions between Humanitarian agents and to convey the planning, trust and reputation information in order to increase the overall help efficiency. To satisfy this goal, the trust model will be integrated with Humanitarian agent's social model and the appropriate subset of the model will be synchronized with stand-ins representing this agent. These agents will provide the others with the necessary information or make commitments on owner agent behalf.

4.4.2 Humanitarian Agents

Humanitarian agent is a special type of agent representing humanitarian relief organization. Normally, it is activated *outside* of the disaster area when the catastrophe strikes.

These agents use their own stock of resources to provide missing resources for the disaster area, compensating the disabled local production. However, to complete this task, they must collaborate with Transporter Agents to deliver the needed goods. Unlike the Location Agents, they don't content



Fig. 4.6. Disaster affects the accessibility in the disaster area (West).

themselves to acquire the transport from the ad-hoc coalition formed by transporters – they perform the coalition leader role themselves, in order to exercise better control over the individual transporters and also to reduce the risk and improve the efficiency using the trust-based planning as described in Section 7.3.

Furthermore, as we suppose that most communication lines are out of order in the disaster area, Humanitarian agents must deploy stand-ins (see Chapter 6.1.4) near the locations requesting help to receive orders and to transmit them to their respective humanitarian agents. Stand-ins are also in charge of the surveillance of the aid delivery - they account how much aid was really delivered to the intended locations and inform their owners about the result.

Humanitarian stand-in is the stand-in agent deployed by the humanitarian agent in the disaster area in order to obtain the information about the needs and results of the operations that are unobservable by the Humanitarian Agent itself. At first, once the disaster is detected by the humanitarian agent, the stand-ins are deployed to the disaster area through path-accessible Location containers, as well as through Vehicle containers (see Figure 4.7). Once deployed, the stand-ins register with local directory service agent as Location agents, so that the Location agents that are in their vicinity can include them into their request queries. Each query that the stand-in is able to cover is passed to the owner, Humanitarian agent, through the synchronization techniques described in Section 6.1.4.

The second role of the stand-in agent is to observe the deliveries of the cargo as it was planned and contracted by Humanitarian agent. To do so, they receive the information about the tasks to observe once the tasks are assigned to individual coalition members and these members have committed to them. Using this information, stand-ins subscribe with the accessible target locations of the relevant cargo batches to receive the notification when the batch (or its part) is delivered. This information is aggregated and passed to other stand-ins, further aggregated and finally it reaches the Humanitarian agent, where it is used for re-planning and trust model update.

Humanitarian stand-in is a domain-dependent example of inaccessibility solution. It is efficient, can adapt to diverse and dynamic accessibility states in the environment and can communicate with Local agents to gather valuable knowledge. Its main advantage in the adversarial environment is the

fact that it can be trusted by its owner¹ – this is a necessary prerequisite, as it uses highly sensitive information about specific task assignments to report their status to the owner.



Fig. 4.7. Humanitarian agent stand-ins are deployed in the disaster area, both on vehicles and in location containers.

Transporter Agent Modification

The original *Transporter Agent* is extended in the scenario in order to be able to cooperate with Humanitarian Agents and to work reliably in the inaccessible and adversarial environment. We have introduced several modifications:

- Ability to act as a coalition member in the mechanism described in Section 7.3 – note that this doesn't include any mathematical programming ability, but simply own resource allocation and team-wide negotiation.
- Ability to react to events encountered by drivers in inaccessible areas: this feature is implemented by placing **Transporter Stand-In Agents** on all own vehicles to handle these events instead of inaccessible transporter agent and to notify the owner directly or through peers as soon as possible/necessary (see Figure 4.8). This approach is preferable to simple Driver agent modification as it allows us to separate the researched mechanism (stand-in operating in inaccessible environment with sparse mobile nodes) from the simulation code in the Driver Agent. It shall be noted that the characteristics of the Transporter stand-in network are different from the one formed by humanitarian agent stand-ins - the transporter stand-ins are deployed on much smaller number of nodes and will not be able to ensure path accessibility between nodes in most cases.

Furthermore, the Driver agent and Location agents were also subject to minor modifications in order to integrate with new or updated agents.

4.4.3 Simulation Agents

In order to simulate the disaster in a realistic manner, we are forced to introduce new functionality into the simulation agents. Instead of creating an additional, specialized agent, we have opted for

¹ In practice, it can be trusted to the extent to which the owner trusts the platform it runs on.



Fig. 4.8. Transporter Agents deploy their stand-ins on their own vehicles only.

the evolution of two existing agent - the Weather agent and the Accessibility agent to achieve the same effect more efficiently.

Weather Agent in its original form provides the weather information that influences the output of production of several commodities, while the others are mere transformations of the primary products. Therefore, when we want to simulate the shortage of goods, the Weather agent is a natural place for the functionality. Each disaster is described by following data: StartTime, EndTime, Area Affected (list of locations) and finally a production reduction coefficient for each affected commodity. Simulated disasters are pre-configured and loaded from configuration file to ensure uniformity between several runs of the same experiment. Each disaster also triggers a topic broadcasted within the master container, that is captured and handled by Accessibility Agent.

Accessibility Agent Evolution is necessary both to simulate the local accessibility faults in a disaster area and the part of the functionality is also necessary to support the experiments with stand-in network adaptation as described in Section 6.2.3. The new functionality, shown in Figure 4.9 includes:

- Local accessibility changes, when the affected area is defined by configuration file and can vary between runs if necessary.
- Handling of the disaster message from weather agent and reducing the accessibility in the affected region.

4.5 Conclusion

Development of the ACROSS scenario has enabled us to study large-scale distributed multi-agent systems with minimal cost. It has been enabled by the high efficiency of the **A-globe** platform, that was further optimized in the course of this project. Currently, as we are still relying on this scenario to gather experimental results, we intend to use **A-globe** to support even more realistic operations and progressively increase their complexity to study more real-world phenomena. This approach allows us to verify the validity of many assumptions that are currently taken as granted in multi-agent research and address potential scalability problems that may become obstacles for system deployment.

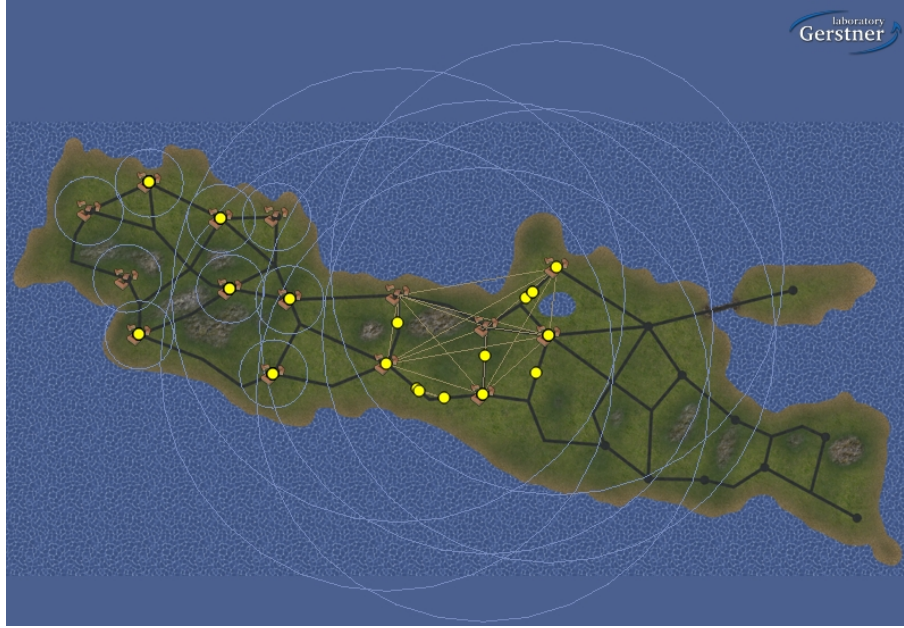


Fig. 4.9. Accessibility agent manages the accessibility to reflect the disaster occurrence, causing inaccessibility within disaster area. Stand-in agents are tested in this environment to ensure knowledge transfer.

5

Using Trust for Coalition Formation in Adversarial Environment

In adversarial environments, agents need a model representing their relationships with other agents, including the measure of adversariality. One of the approximations of this measure is *trust*, discussed in this chapter.

The problem of trust in multi-agent systems is already a relatively well defined one, with many important contributions in the area. However, there are many approaches to trust representation, using different scales and various decision and learning algorithms. In this chapter, we are going to analyze the existing formalisms of the trust representation and propose their extension with explicit uncertainty representation using fuzzy numbers and fuzzy set theory.

To differ from the current work in the domain, we have decided to devise a model with features that are crucial for embedded applications with minimum or no human control. Therefore:

- we don't concentrate our attention on a simple client-provider relationship, but we examine a more general coalition case in which the potential coalition members must select their partners carefully, as the failure of any partner in the coalition influences the payoff of all members.

- We have concentrated our study on general, not situational trust, as the autonomous devices provide specialized services in most cases. On the other hand, our experiments in the last chapters show that the situational trust can benefit even such devices, and remains in the scope of our future work.
- Autonomous adaptation to previously unknown failure rate in the environment makes the model robust and minimizes the human manipulation necessary beforehand, as well as the amount of the information necessary.
- Model is robust with respect to significant environmental noise, as shown in experiments below.
- Model is computationally efficient and lightweight. Knowledge is updated iteratively and necessities only several bytes of storage per each evaluated agent.

5.1 State of the art

Existing literature on trust in MAS is abundant. Basic definitions and terms seem to be well defined by works of Marsh [43] and Castelfranchi et al. [16]. Marsh defines the general trust - trust in an agent as an individual, as: "Agent x expects that y will behave according to x 's best interests, and will not attempt to harm x ". This is the definition we have chosen for the current version of the formalism and the set Θ , defined in section 5.4 is defined accordingly. Castelfranchi provides an important distinction between trust as a *mental state*, measure of agent's trustfulness and the trust as a *decision to delegate* (or more precisely to cooperate, when the delegation is replaced by cooperation) where the mutual trust between each pair of agents in the coalition is necessary.

The trust learning has been well explored by Andreas Birk et al.[10], T. D. Huynh et al. [38] and others. Very good formal model has been recently contributed by S.D. Ramchurn et al.[59].

Use of the trust for partner selections has been analyzed by Marsh [43], T. Dong Huynh et al [38] and others [32].

Considering the trust as a mental state, we have several options how to represent it. The first option available is a simple binary value - agent is either trusted or not. This option makes a trusting decision very easy, but comes with a price of low information content and no option to store the previous interaction history. Typically, the use of this trust representation corresponds with tit-for-tat strategy [6] from the iterated prisoners dilemma. In the real life, this strategy also suffers from the difficulties in deciding whether a result of previous action was a success or not. In many real environments there may not be a simple binary answer, as the environment and context influence the results of all agents' actions and the definition of success itself may be ambiguous.

We may improve this simplistic binary approach by using a real number value for the trust, typically from the interval $[0, 1]$ or $[-1, 1]$. This approach has been used by most researchers so far.

The issue of the interval used for the trust value is not crucial, as a bijection exists and simple techniques allow us to map the representation in one interval to another. However, in reality, this issue influences the use of the trust values in agents decision making process. When we use the interval $[0, 1]$, the probability analogy comes into the play naturally. The probability interpretation makes the use of the trust quite straightforward, using the game or decision theory reasoning processes about expected utility. In the same way, it simplifies the learning process, as the ratio of expected and achieved utility in previous interactions with the trusted agent provides the trustor with an easy trust estimation.

The other approach, using the interval $[-1, 1]$ emphasizes the cognitive aspects of the trust - 1 means that the agent is completely distrusted, value 1 means completely trusted agent. Appropriate methods, for example Fuzzy cognitive maps [17], known from the expert systems field, are appropriate for handling of the trust and can partially or completely solve the problems, caused by naive trust use, that were identified by Marsh [43].

However, both approaches suffer from one fundamental flaw - it is very difficult to distinguish between the ignorance and unpredictable behavior. The fact that both cases are typically represented

by values around 0.5 or 0 in respective cases makes the meaningful use of the central part of the scale quite challenging. Moreover, the fact that the uncertainty is represented by a specific value or range makes difficult the adaptation to the environments with various levels of noise in the communication or action results. Informally, the trustfulness considered as low in one environment may be found sufficient in different context.

Furthermore, in much of the previous work the trust between two partners with well defined client and supplier roles was examined. In our experiments, we focus on the issues of trust in peer-to-peer coalition cooperation, when the failure of cooperation may be attributed to the fault of one or more coalition partners, or to the environment effects (noise).

Therefore, we argue that the existing formalisms for the representation of the trust are insufficient, as we still confound uncertainty and the level of trust itself. For us, trust is not a single value, but rather a complex mental state that includes the uncertainty.

We were partially inspired by the work of Ramchurn et al.[59], where a fuzzy set based approach to trust categories is already proposed, even if the value representing the trust is still crisp. In our work, we further extend the approach with the mathematical apparatus known from the field of fuzzy set theory [54] and fuzzy control [22] by the introducing the fuzzy numbers.

But before we specify the representation we use, we shall present the criteria that has led us towards the selection of this concept.

5.2 Requirements on Trust Representation

We propose to enrich the existing trust representations presented in the previous section with uncertainty. This is not an easy task, as the resulting formalism will be undoubtedly more complex than simple real values and we shall be able to select a well adopted formalism from a vast choice available - the possibilities range from probability distributions on one side to possibility theory and Dempster-Shafter on the other. To make a well adopted choice, we propose following requirements for the trust representation formalism:

- Trust in agents shall be comparable - trustfulness functions shall form a lattice.
- Trust in agents shall be comparable with intuitively predefined threshold. The signification of this threshold shall be understandable and the threshold must be easy to define.
- The signification of trustfulness value shall be understandable.
- We shall be able to deduce the trustfulness of the group of agents from the trustfulness of individuals in the group. At first, we may ease this requirement by assuming that the trustfulness of an individual is not influenced by other members of the group.
- Trust shall be dynamic and adaptive. It shall be possible to derive it from previous interactions.
- The trust learning process shall be iterative and reasonably fast and lightweight.
- Besides learning, agents shall be able to include arbitrary beliefs about other agents into the trust.
- Trust shall support the integration of reputation information as received from other. It must be usable as a source for the reputation information provided to others.
- Membership in social structures (coalitions, alliances) shall be expressed by the same or similar formalism. It shall be able to define a relation between trust and the membership values.
- Extension towards situational trust shall be possible in the future. This means including more inputs, besides apriori general trust and storing rules determining the agent's behavior in different situations.
- Learning of these rules shall be possible, with or without background knowledge.

Using the criteria presented above, we have selected the fuzzy numbers approach. We have not opted for the used of probabilistic methods due to their relative heavy weightiness. We have also considered that the very strict rules of statistic decision can make the passage towards the situational

trust difficult as the usefulness of the approach decreases with increasing number of dimensions in the problem. On the other hand, we felt that the possibility theory based approach is not sufficient, as it fails to provide a single significant value representing the trust - a crucial feature for comparisons and easy understanding.

The fuzzy number approach provides a well balanced compromise between the information content and the ease of use, as was already proved in the booming field of the fuzzy control. One of the important factors in our decision was the fact that the existing family of fuzzy controllers proves that the approach can be successfully integrated with restrained hardware platforms. This is an important advantage for the use of this methodology in pervasive computing field.

Solid mathematical background [15] exists also in the domain of mathematical programming and decision-making with uncertain information represented by fuzzy numbers. Therefore, trust represented as a fuzzy number can be easily integrated with advanced planning algorithms, as shown in Chapter 7.

5.3 Fuzzy Numbers

Fuzzy numbers are extension of normal, crisp numbers in the same fashion as the fuzzy sets are the extension of crisp sets. We may define (see [23]) the fuzzy number as a *normal convex fuzzy set on the real line*, where *normality* means that the height of the set is 1; i.e. set has a non-empty core. The set is said to be *convex* iff $\forall(x, y, z) \in (R^3), x \leq y \leq z$ holds that $\mu(y) \geq \min(\mu(x), \mu(z))$. Example of fuzzy number values can be found on fig. 5.1 or fig. 5.2.

Informally, the *core* of the fuzzy set is defined as a subset of the fuzzy set containing the elements x whose membership $\mu(x) = 1$. In some definitions, it is required that the fuzzy number core shall be a single value and when this condition is not fulfilled, the term *fuzzy interval* is used. In this text, we will only use the term fuzzy interval to emphasize the cases when the fuzzy number represents a range, rather than value, but all results relevant for fuzzy numbers are valuable for fuzzy intervals as well.

As we represent the trust, we restrict the support of the set to the $[0, 1]$ interval. Moreover, we limit ourselves to the fuzzy numbers defined by piecewise linear membership functions on the above specified interval, to speed-up the computation and inference process.

However, the extension of simple real values to fuzzy numbers necessities also the extension of basic operations on these quantities.

Ordering and ranking is a crucial operation necessary for partner selection. It has been studied for example by Fortemps and Roubens in [29] or in [12]. The authors provide a method that is very easy to implement with the limitations we have adopted and gives us the results that are intuitive enough. The most notable difference from the real numbers is that the antisymmetry does not hold for this relation on fuzzy numbers.

While using the fuzzy rules to take trusting decisions, we use the fuzzy numbers as inputs for these rules. Therefore, we must be able to determine the **inference** with fuzzy intervals representing the rules/decisions. We have opted for the use of Mamdani inference, defined as

$$D_T(X^*, A_i) = hgt(X^* \cap_T A_i) \quad (5.1)$$

where T is a selected *t-norm* (see [54]). We have selected the Standard (Gödel, Zadeh) t-norm, defined as

$$T(A, B) = \min(A, B) \quad (5.2)$$

In theory, all the agent's qualitative reasoning about the cooperation could have been done using the fuzzy numbers and appropriate fuzzy arithmetics operations. In practice, the complexity of this approach could not be justified in most cases. Therefore, **defuzzification** of trust values is necessary

for selected operations. An obvious choice for defuzzified value is the core of the fuzzy number. In the case of the fuzzy interval, the center of the core is chosen.

Fuzzy numbers presented in this paragraph are used in the formal model we propose to represent the trust in agents.

5.4 Formal Model

In our formal model, we extend the existing trust representations using the fuzzy set theory. To do so, for each agent A we define a set of agents trusted by agent A , denoted Θ_A . We denote $\Theta_A(X)$ the membership function of this set defined on the set of all agents known to the agent A .

Whether Θ_A is a fuzzy set or not depends on the value range and type used for trust definition. Binary trust mentioned above results in a normal, crisp set - membership function takes only two values, $\Theta_A : Agents \rightarrow \{0, 1\}$ - agent is either trusted completely or not at all. Use of the real value in the $[0, 1]$ (or $[-1, 1]$ with transformation) interval defines a standard fuzzy set, $\Theta_A : Agents \rightarrow \{[0, 1]\}$.

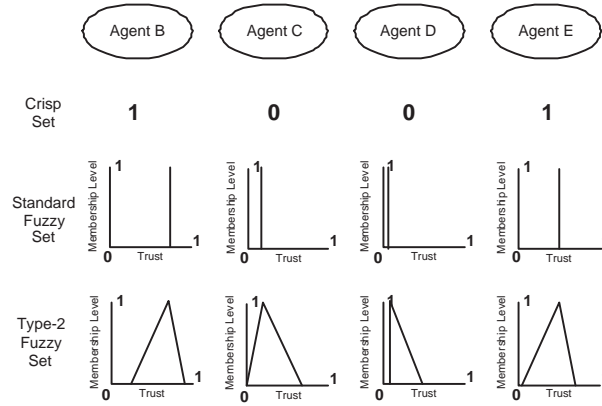


Fig. 5.1. Comparison of membership values for three types of the set - a crisp set, a standard fuzzy set and a type-2 fuzzy set with fuzzy number membership.

Use of the fuzzy numbers to represent trust makes the set Θ_A a type-2 fuzzy set, as the membership function itself is a fuzzy set, albeit a simple one (see for example [54]). This does not pose any serious problem to us, as the mathematical concepts necessary to work with fuzzy values are already well developed in the fuzzy control field. All the variants are shown in fig. 5.1.

The set Θ_A represents the agent's A trust in other agents as a mental state. Besides this representation, we need to address the following problems: (i) deriving the trust values from the experience, (ii) updating the trust in agents using these values and (iii) using the trust values to make cooperation decisions. We will address these issues in the following sections.

5.4.1 Deriving Trust Observations from Coalition Cooperation Results

In this section, we will propose a general method how to evaluate the trustfulness of the coalition partners in a specific coalition C as a function of the utility generated by the cooperation. Using this method, each coalition member A can obtain a single value in the $[0, 1]$ interval representing the trust observation τ for each coalition member $Agent$, denoted $\tau_{C, Agent}^A$ or simply $\tau_{C, Agent}$ when no confusion is possible.

We have decided to use a completely peer-to-peer approach that can be applied in most environments where the agents cooperate to achieve their goals. As we try to keep our algorithm as domain independent as possible, we start by normalizing the cooperation result into $[0, 1]$ interval. The simplest way would be to use the minimum utility (maximum loss) u_{min} , expected (maximum) utility u_{max} and real, obtained utility u to measure the *success ratio* u_n using the formula

$$u_n = \frac{u - u_{min}}{u_{max} - u_{min}} \quad (5.3)$$

In theory, we could stop here. In practice, this linear relation is rarely appropriate, as shown by many experiments [58]. Therefore, we will typically replace this relationship by subjective loss function with u_{min} and u_{max} as parameters and u as an input to model the perceptions of gain or loss by the agent. The result of this transformation is called *subjective utility* u_s^A (or simply u_s).

As a simple example of such function, our agents do perceive the losses worse than linearly. Therefore, they obtain their final *subjective utility* by raising the value u_n to power of two, obtaining the value $u_s^A = u_n^2$ used as an input for the suite of the process.

Raising to the power of two is an arbitrary choice, modelling the fact that the losses are perceived worse than their real value, but may have another signification - attribution of the blame to an individual, rather than to some stochastic process [35].

Each coalition member calculates its value u_s^A and uses this value to obtain the values $\tau_{C,Agent}^A$ for all other coalition members. Different strategies may be used to do so, analogously to profit distribution in coalitions[40]. The cases we consider in the scope of the current work are:

- **Equally** - the value u_s^A is used as an input to update trust in each coalition member.
- **Proportionally** - the observation value depends on the defuzzified apriori trust the agent has in the coalition member and the u_s^A . Currently, we use the relation:

$$\tau_{C,Agent}^A = \frac{defuzzy(\Theta_A(Agent)) \times u_s}{Avg_{Agent_i \in C}(defuzzy(\Theta_A(Agent_i)))} \quad (5.4)$$

where $\tau_{C,Agent}$ denotes the trust observation (real number) we derive from the agent's participation in the coalition C . $defuzzy(\Theta_A(Agent))$ denotes the center of the apriori trust membership function, formally core of agent's membership function in Θ_A .

The value u_n^2 (or $\tau_{C,Agent}$ respectively) is then used as an input for updating the trust in *Agent* as described in the following section.

5.4.2 Iterative Learning of Trust Values

When we try to represent the trust in agent B $\Theta_A(B)$ as a single fuzzy number, we must be able to find an optimum form of this number to represent the past experience. Formal restrictions on the fuzzy number are not very strict, but we limit ourselves to *piecewise linear fuzzy numbers* and *iterative* learning in order to keep the processing lightweight and well adopted for potential embedded solutions.

To simplify the notation, we will denote τ_B^A or τ_B all trust observations of agent A about the agent B - suite of n_B real values in $[0, 1]$. Note that these values are not kept in agent's memory, as the learning is iterative.

Strictly speaking, iterative learning requires that the new value for the $\Theta_A(B)$ is obtained only using the apriori value of the $\Theta_A(B)$ and the observation $\tau_{C,B}$. In order to make our algorithm easy to understand, we shall maintain some supplementary simple data - like n_B , number of previous observations used to establish the trust in agent B and the $Avg\{\tau_B^2\}$ used to estimate the variance of the data as shown further.

Trust *average* $Avg\{\tau_B\}$ is a good candidate to define the center of our fuzzy number - it is easy to understand, it is consistent with non-fuzzy approaches and it may be computed iteratively, provided that we keep both the current average and n_B - number of observations used to obtain this value.

But besides the center, we need to represent the uncertainty at least by specifying the left and right sides of the fuzzy number membership function. The simplest way to do so is to use triangular fuzzy number and to define left and right bounds as $\min\{\tau_B\}$ and $\max\{\tau_B\}$. This corresponds indeed with the intuitive interpretation using the possibility theory (min and max values provide us with an interval where the whole past experience falls). Moreover, the values are trivial to maintain iteratively. On the other hand, two major drawbacks outweigh the advantages of this method - (i) both values tend to have only limited information content in the real environments, as they are often very close to 0 or 1 for most agents B and (ii) the uncertainty of the trust is actually non-decreasing with growing number of observations n_B , what is being counterintuitive. Another value,

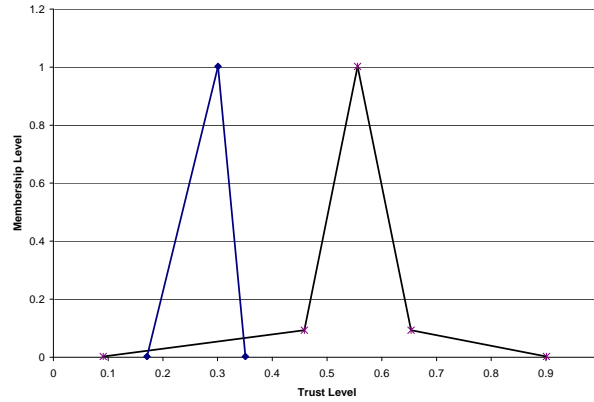


Fig. 5.2. Two proposed fuzzy number shapes for the trust representation - simpler triangular form on the left, complex one on the right. Each represents different trust value.

at least partially solving the issues with minimum and maximum approach, is *variance* $\sigma_A^2\{\tau_B\}$ or *standard deviation* $\sigma_A\{\tau_B\}$. It also has an advantage of being well understood and even if it cannot be maintained iteratively as easily as average, it is possible to estimate it rather exactly using the well known relation:

$$\sigma^2\{\tau_B\} \leq \widehat{\sigma^2\{\tau_B\}} = Avg\{\tau_B^2\} - Avg^2\{\tau_B\} \quad (5.5)$$

With growing number of observations, the right-hand side of the inequality approximates the variance fairly well.¹

The first representation we propose is a simple *triangular fuzzy number*, defined by three points and two straight lines. The center - core of the fuzzy number - $defuzzy(\Theta_A(B))$ is defined by the average $Avg\{\tau_B\}$, left hand-boundary by the value $\max\{\min\{\tau_B\}, Avg\{\tau_B\} - \widehat{\sigma_A\{\tau_B\}}\}$ and the right-hand boundary analogously by the value $\min\{\max\{\tau_B\}, Avg\{\tau_B\} + \widehat{\sigma_A\{\tau_B\}}\}$. Note that even if the set τ_B is formally used in the definition, the min, max and other characteristic values we actually use can be updated iteratively and no explicit history representation is used. Due to the cropping of the number by the min and max values, this representation does not necessarily define symmetric numbers.

Second representation that we propose extends the previous one by the inclusion of the min and max data directly into the fuzzy number shape. The left boundary is now defined as $\min\{\tau_B\}$, right boundary as $\max\{\tau_B\}$, both with the membership = 0. Core is defined by the average value

¹ We assume that the trust observations are conform to Gauss distribution.

$defuzzy(\Theta_A(B)) = Avg\{\tau_B\}$. However, two points were added on each side of the center, both with membership $= \frac{1}{n_B+1}$. Their coordinates are the same as for the left and right boundary in the simpler version: $max\{\min\{\tau_B\}, Avg\{\tau_B\} - \sigma_A\{\tau_B\}\}$ and $\min\{\max\{\tau_B\}, Avg\{\tau_B\} + \sigma_A\{\tau_B\}\}$. This shape has an advantage of representing better the current experience and the uncertainty decreases relatively steeply with the growing number of observations - the property that we consider crucial for future extensions of the model. Both definitions are compared in fig 5.2.

In order to support the recent model developments towards better integration with social models and planning and reasoning algorithms, two new concepts are introduced into the trust model. **Distrustfulness** is defined as $\Delta_{A_j}(A_i) = 1 - \Theta_{A_j}(A_i)$ - it can be derived from trustfulness and is only defined as a shorthand for easier problem specification. In its definition, we use classical definition of fuzzy number subtraction $(A - B)(y) = \sup_{x_1 - x_2 = y} \min\{A(x_1), B(x_2)\}$.

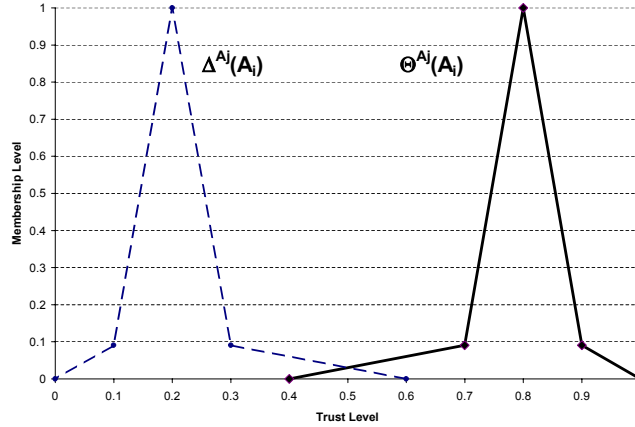


Fig. 5.3. Shapes of trust $\Theta_{A_j}(A_i)$ and corresponding distrust $\Delta_{A_j}(A_i)$ function.

Action Trustfulness

In addition to the above concepts related to agent trustfulness, we may use the weak delegation concept as introduced in [16] to define the trustfulness of non-agent elements that we rely on without the explicit commitment from them. In our case, we define the trustfulness of abstract action: Θ_{a_k} represents a *trustfulness of a particular action* a_k and not of an agent². Regardless of this fundamental difference, the approach to its modelling remains strictly the same as for the agent trustfulness and the update mechanism is still the one described for agents. Once we terminate the plan execution and evaluate the new trustfulness values, we may determine the trustfulness of actions in the same manner as trustfulness of individual agents, as each action contributes to the global goal achievement. In the simplest manner possible, the agent may assign the same value to all actions, effectively disabling the "overbooking" mechanism in relation 7.1 - such behavior is appropriate for environments with low failure rate. If the agent opts for action-level modelling, it may either consider the actions on the same level as agents (action is considered as a team member for the purposes of trust modelling), or it can model them on higher level and use the output of this level to determine the expected outcome that is then used for agent's trust modelling.

² See Section 7.3 for more details on actions.

5.4.3 Self-Trust as a Parameter for Trusting Decisions

In the paragraph above, the agent has never excluded itself from the group of agents between whom we distribute the collaboration trust value. It means that the agent actually estimates the trust in itself: $\Theta_A(A)$. There are two good reasons for such behavior.

First, an agent does not necessarily trust itself - we may easily imagine a situation when an agent runs on a hardware with malicious intruding software and is almost never able to protect its private data and communications from platform-originating intrusion [4]. When the agent observes its self trust and detects a significant decrease, it may decide to migrate, to interrupt communication with others or even to terminate itself in order to protect the cooperators.

The other reason why we measure the self-trust is environmental adaptation. In many cases, it is difficult or even impossible to estimate correctly what is the expected payoff of the cooperation in the given environment. In our approach, we don't take this factor into account during the evaluation of the cooperation success - we rather integrate this information into the cooperation rules derived from the self-trust data.

We define two linguistic variables [22] on the trust membership support $([0, 1])$. First of them is a *low trust* domain, denoted LT^A while the other is *high-trust* domain, HT^A . The sum of their membership functions is equal to 1 on the whole interval $[0, 1]$ - they form a partitioning of unity.

We use the self-trust data to establish the fuzzy intervals HT^A and LT^A as follows. First, we define that $HT^A(1) = 1$, a natural assumption as the complete trust is undoubtedly high. Then, we say that agent A considers itself as trusted. (We use the self-trust for environmental adaptation, rather than for intrusion detection.) Therefore, we say that $HT^A(\text{defuzzy}(\Theta_A(A))) = 1$. From this value on, we decrease the trust linearly until we reach the 0 membership for the trust = $\max\{\min\{\tau_A\}, \text{defuzzy}(\Theta_A(A)) - \sigma_A\{\tau_A\}\}$. LT^A is complementary - it is equal to 1 between 0 and $\max\{\min\{\tau_A\}, \text{defuzzy}(\Theta_A(A)) - \sigma_A\{\tau_A\}\}$, where it starts to decrease linearly to finally reach zero at $\text{defuzzy}(\Theta_A(A))$. The use of linguistic variables for the inference process is shown in fig. 5.4.

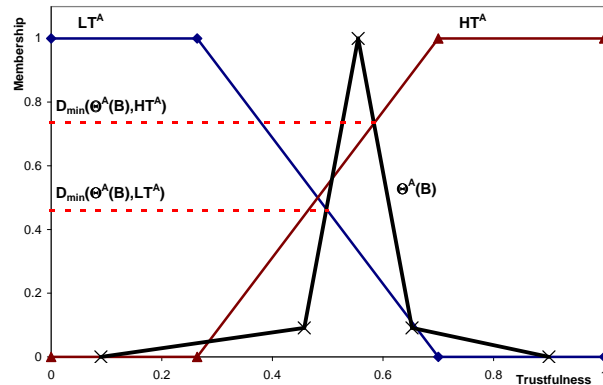


Fig. 5.4. Example of the trust decision. Height of the intersection with low trust (left segment) and high trust (right segment). As the incidence with the high trust is bigger, agent is considered to be trustful.

5.4.4 The Decision to Cooperate and Partner Selection

A set Θ_A and the fuzzy intervals HT^A and LT^A represent the mental state of the agent.

When an agent proposes a coalition or is invited to participate in one, it needs to take a trusting decision; it has to decide which other agents are admissible as partners and order the admissible partners by trust to minimize the risk.

To establish whether an agent B is trusted, we use the formula 5.1 to calculate the incidence of the trust in the agent B with the intervals HT^A and LT^A .

$$D_{min}(\Theta_A(B), HT^A) = hgt(\Theta_A(B) \cap_{min} HT^A) \quad (5.6)$$

$$D_{min}(\Theta_A(B), LT^A) = hgt(\Theta_A(B) \cap_{min} LT^A) \quad (5.7)$$

Agent B is considered to be *trusted* iff $D_{min}(\Theta_A(B), HT^A) \geq D_{min}(\Theta_A(B), LT^A)$.

When an agent A needs to organize a coalition, it identifies a subset of trusted agents. Then, it calculates the *usefulness* (see [16]) of these agents for the coalition using the social knowledge in its acquaintance model. The usefulness of each agent is then multiplied by the trustworthiness (defuzzified) of this agent, to account for the *willingness*³ and the candidates are ordered by this value. Suitable subset of acceptable candidates is then invited to form a coalition.

In the opposite case, when the agent A is invited to participate in a coalition, it evaluates its trust in the members of the coalition. When all members are considered to be trustful, it agrees to take part in the coalition.

5.4.5 Trust Module Overview

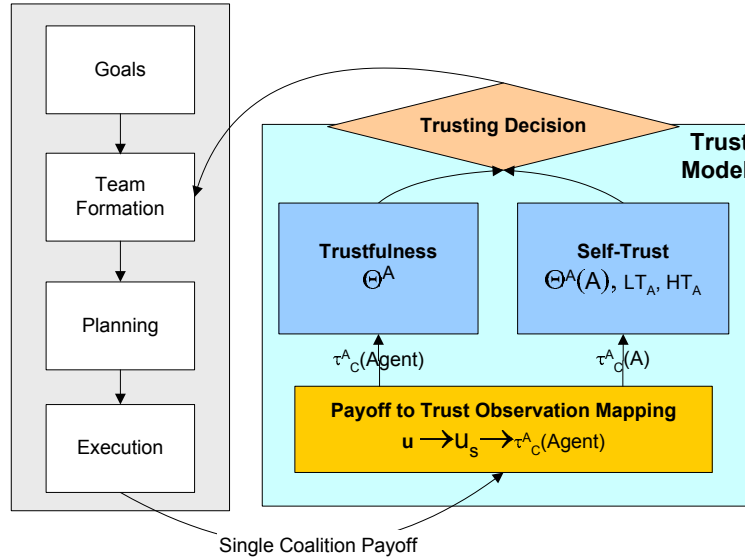


Fig. 5.5. Model structure and integration with agent reasoning.

In Figure 5.5, we can see how the model integrates with the rest of the agent reasoning (left column) and how its components mentioned above are updated and used to take decisions. To recapitulate, the utility function is used to provide relevant, but normalized input for trust model when the coalition cooperation ends. The data is normalized and split between coalition members, including the evaluating agent, for which we modify the self-trust value. This value is then used to characterize the environment in which the agent operates, making the environment recognition automatic. When the next team is being formed, the trustfulness of each member and leader is evaluated and agent takes decision whether to participate/invite or not.

³ In this work, we consider usefulness and willingness as perceptions. Therefore, we model the environment effects in the willingness rather than in the usefulness part, that is established using the information provided by agents themselves.

5.5 Experiments

In this section, we are going to present the empiric evaluation of the general trust we present.

The crucial quality of our model is robustness with respect to various levels of failure in the environment and autonomous adaptation to current levels of failure. In the experiments shown below, we show that the same model, without any pre-configuration can be applied in the environments ranging from situations where the failure is exceptional to the situations where the success is rare. In all the environments evaluated, we observe the speed of the adaptation - intuitively, the model adapts best in the easiest situations and takes more time in the difficult settings with overwhelming background noise.

5.5.1 Configuration of Experiments

In all our experiments, based on the ACROSS scenario described in Section 4.3, we have used equal blame distribution between coalition members. The two groups of agents for which we present the results consisted of 9 fair agents and 1 defector and 5 fair agents and 1 defector respectively.

In the two series shown in the following sections we evaluate the external effects of the trust on the agent's behavior, as well as the inner working of the trust model. First series, presented in Section 5.5.2 concentrates on the external factors, as we compare the number of coalitions in which the defecting agent participates and we look for the decrease in its coalition potential - a clear proof that other agents progressively refuse any cooperation with this agent. In the second round of experiments, as presented in Section 5.6, we evaluate the trustfulness of defector as observed by other alliance members and compare - we actually look behind the coalition cooperation output and observe the actual trusting decision.

In the first series of the experiments (Section 5.5.2), we evaluate the model in three environments - A, C and D as defined in Table 5.1, while in the second series (Section 5.6) we add an intermediary scenario B. Scenarios differ by the level of background noise (random attacks) with increasing difficulty - background noise increases from virtually nil in the Scenario A to 80% of the data in the Scenario D. Variable r_{attack} denotes the average ratio of informed attack attempts to uninformed ones and $r_{success}$ the average ratio of informed successful attacks (when some goods was stolen) to uninformed ones.

	p_n	p_d	r_{attack}	$r_{success}$
Scenario A	0	0.04	—	—
Scenario B	0.0005	0.04	1 : 1.8	1 : 0.85
Scenario C	0.002	0.04	1 : 7	1 : 2.5
Scenario D	0.004	0.02	1 : 12	1 : 4

Table 5.1. Scenario settings.

5.5.2 Results - Coalitions

In the first series of experiments, we compare the average cumulated number of coalitions for the fair and deceiving agent.

The three membership function representations we evaluate are:

- **Real number** trust, when the trust is represented by a single value $Avg\{\tau_B\}$.
- **Simple sigma** fuzzy number trust, as defined by the first(triangular) description in Section 5.4.2.
- **Enhanced** fuzzy number trust, as also described in Section 5.4.2.

We have conducted nine sets of experiments, where each set consisted in five runs to eliminate random factors. We have chosen three different membership function listed above and we tested them under three different settings regulating the background noise - Scenarios A,C and D as defined in 5.1.

Scenario A.

In the first case, the task was an easy one - the background noise was limited and attacks were possible only in the case when a defecting agent was a part of the coalition. All three representation methods have performed well and the defectors were correctly identified, as we can observe on figures 5.6, 5.7 and 5.8. We may attribute the lag in detection of agents to the length of cooperation and to the need to establish a significant experience in order to refuse cooperation.

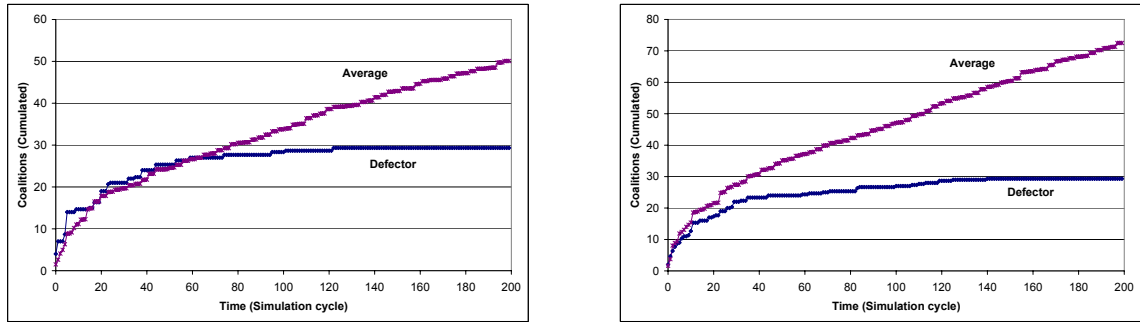


Fig. 5.6. Scenario A: Environment with low level of background noise and real valued membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

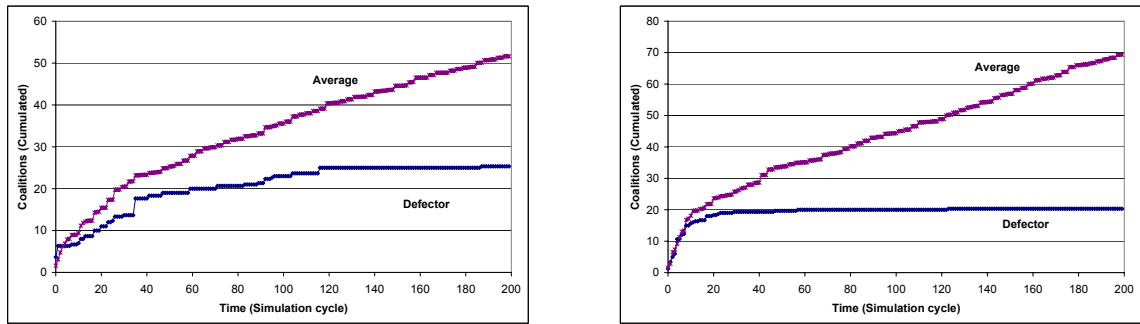


Fig. 5.7. Scenario A: Environment with low level of background noise and simple sigma membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

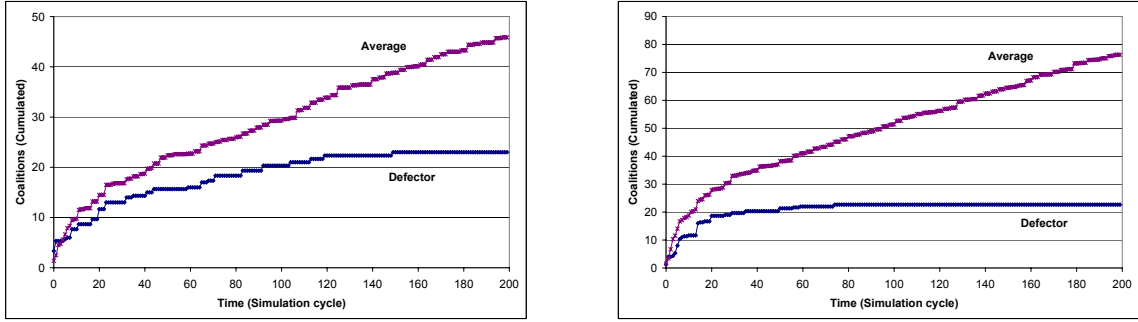


Fig. 5.8. Scenario A: Environment with low level of background noise and enhanced membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

Scenario C.

In the second case, the background noise was increased by activation of random attacks. This made the identification process longer, as the agents need to identify both the environment characteristics (using the self-trust) and the characteristics of their cooperators. Furthermore, the coalitions with and without defector are more difficult to distinguish now. Due to these issues, some of the agents cooperate with defector until later in the runs. However, agent is successfully detected and refused by the majority of coalitions.

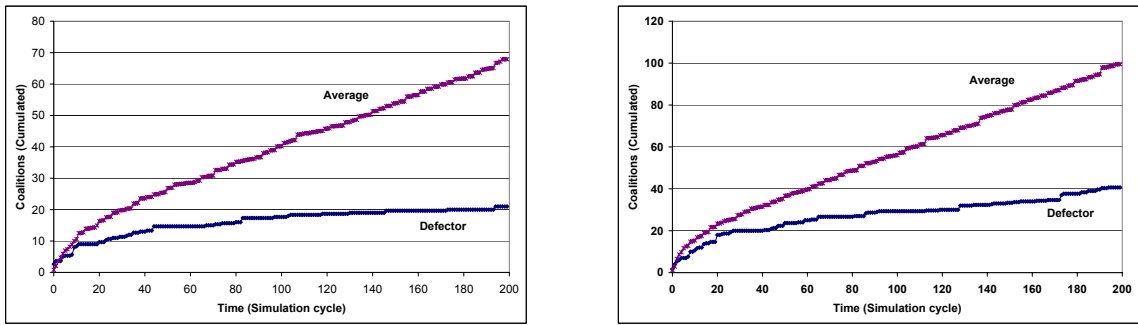


Fig. 5.9. Scenario C: Environment with medium level of background noise and real valued membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

Scenario D.

The third configuration is the most challenging one. We have further increased the background random attacks probability and decreased the value of the information provided by defector by decreasing the p_d value. The results are mixed in this environment and provide a valuable guidance for future enhancements of the model. In the big alliance, agents have no trouble with identification

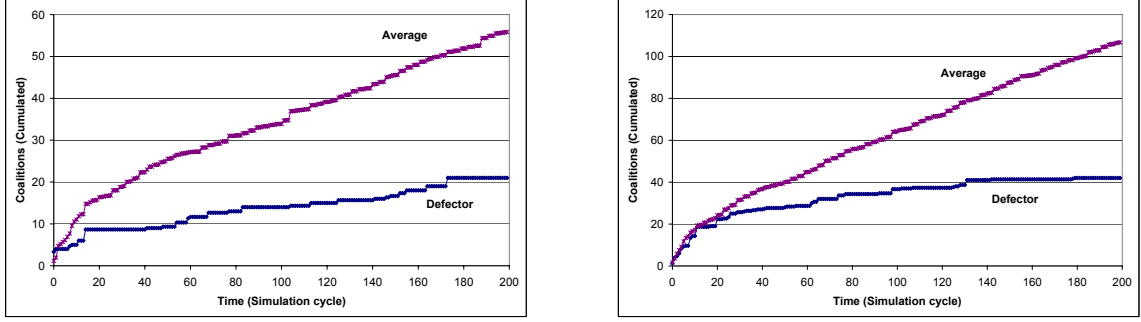


Fig. 5.10. Scenario C: Environment with medium level of background noise and simple sigma membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

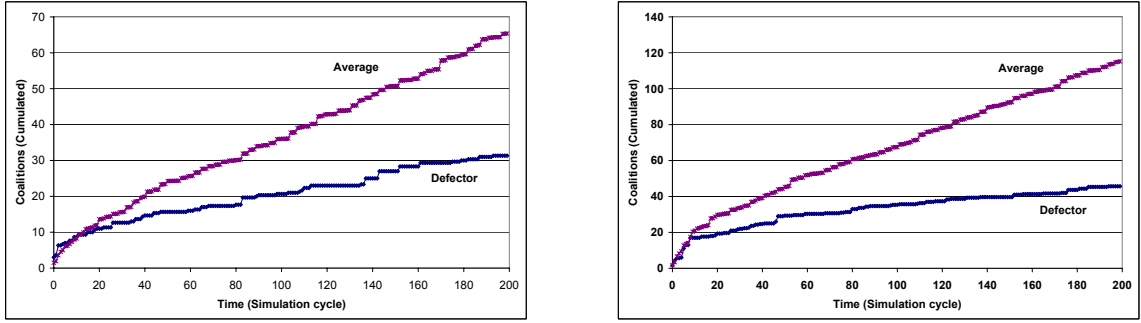


Fig. 5.11. Scenario C: Environment with medium level of background noise and enhanced membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

of the defector, even if the success is not perfect through all runs (averaged on the graph). In a smaller alliance, the agents were unable to eliminate the defector. We may attribute this to the fact that frequent cooperation with defector has influenced the environmental estimation of at least a part of the agents, who therefore judge the defector trustworthy and attribute the failures to the environment.

In the second series of experiments, we will concentrate on the underlying trust decisions that define the above behavior and the overall evaluation will be provided in Section 5.6.1.

5.6 Results - Trust

In the second row of experiments, we have conducted four experiments, in the settings defined by Scenarios A,B,C,D as shown in table 5.1. Results are provided for the bigger alliance only.

In these measurements, we have focused on a trust evolution during time in a noisy environment. Data plotted in the graphs shows for how many agents in the alliance the particular agent is trustworthy. For sake of simplicity we have chosen only values of the defector and the average value

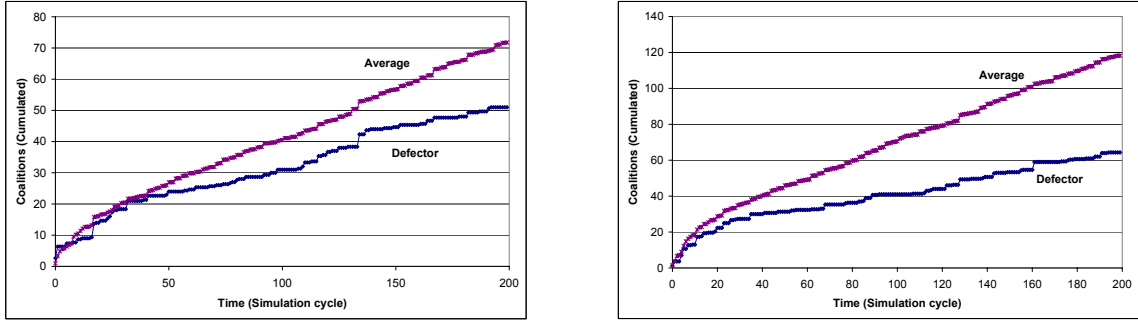


Fig. 5.12. Scenario D: Environment with high level of background noise and simple membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

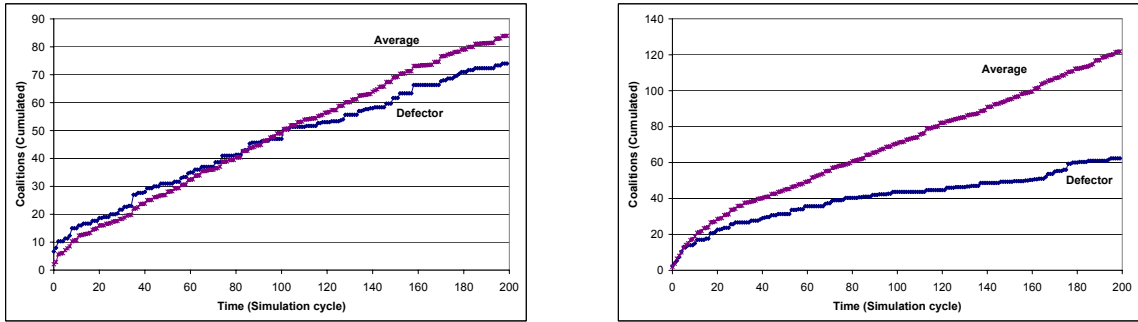


Fig. 5.13. Scenario D: Environment with high level of background noise and simple sigma membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

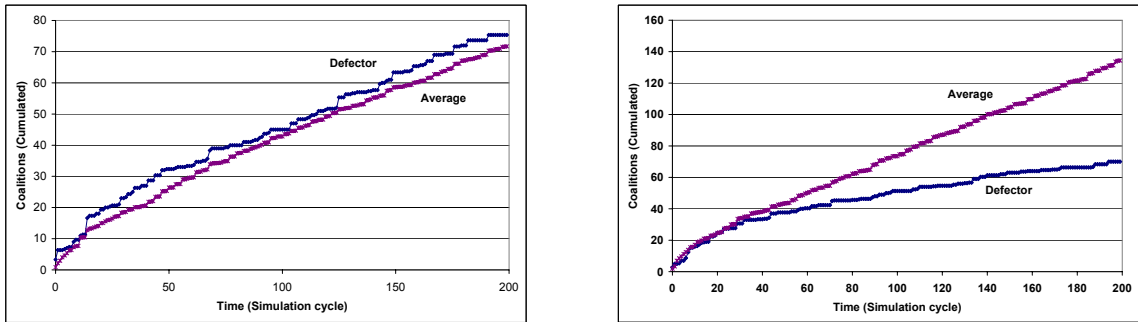


Fig. 5.14. Scenario D: Environment with high level of background noise and sophisticated sigma membership function - *Left*: Smaller alliance. *Right*: Bigger alliance.

throughout the whole alliance, including the defector. Size of the gap between the two values in the given moment t grows with the number of agents that has detected the defector by the time t .

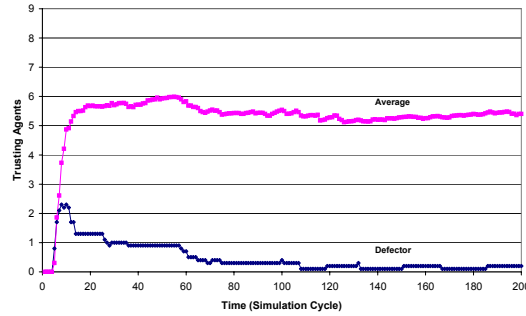


Fig. 5.15. Trust evolution in Scenario A (limited background noise).

As in the previous case, the scenario A is considered to be a reference case as there are no background attacks. We can observe how the trust of alliance members into defective agent falls – figure 5.15. We should mention that average value of trusting agents is lower than in next scenarios because each agent has a high self-trust and is therefore cautious trusting someone else. The actual numerical trustfulness values $\Theta_A(B)$ are lower than in the other scenarios, but the self-trust value correctly identifies the environment as a low-risk one.

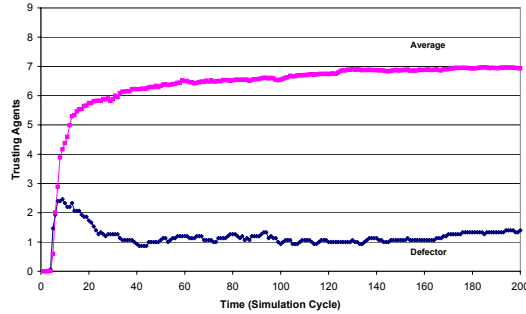


Fig. 5.16. Trust evolution in Scenario B (low background noise).

In the second and third scenario (B,C), a new source of the background noise was introduced by the activation of random attacks. This made the trust learning harder, as the agents need more data to distinguish the coalitions with and without defector. Due to this issue, some of the agents trust the defector and are ready to cooperate with it until later in the runs – figure 5.17. However, agent is successfully detected and refused as a coalition member by the majority of agents, especially in fig. 5.16, where the noise is comparable with the signal. We should emphasize that agents were able to reveal defector from data (C) where only 29% of the attacks were caused by member defection, while the 71% of attacks are attributed to background noise.

In the scenario D we have further increased the background noise and simultaneously decreased the probability of attack with defection, making the environment even harder. As graph in figure 5.18 indicates, settings for Scenario D are on verge of reasonable limits. Although we can still detect the defector, the difference between defector's trustfulness and average trustfulness value in alliance is low - about 10%. In such setups, we discover the limits of our approach - only about 20% of the

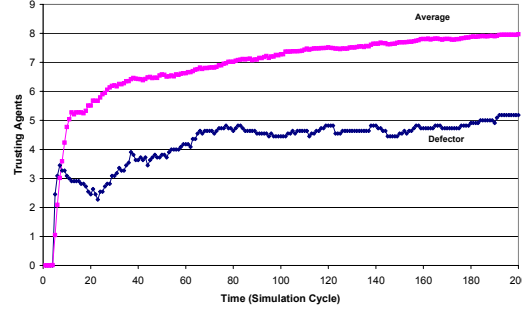


Fig. 5.17. Trust evolution in Scenario C (medium background noise).

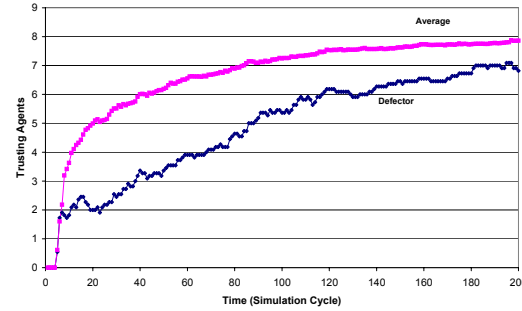


Fig. 5.18. Trust evolution in Scenario D (high background noise).

observations correspond to the real agent defection. In the real applications, we can easily improve algorithm performance by inclusion of context information (for instance the transport path length in our case) and improve the algorithm performance in the well defined context.

The final graph (figure 5.19) summarizes all the results so that they can be compared.

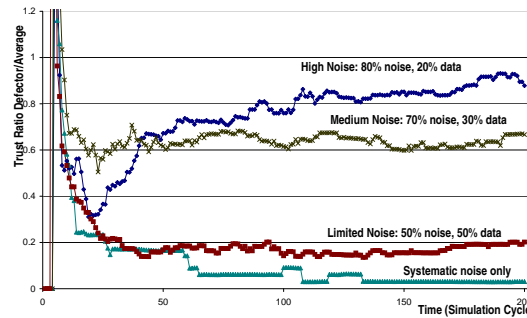


Fig. 5.19. Comparison of results between scenarios with different levels of noise. Lower values are better.

5.6.1 Observations

The preliminary experiments we have realized and described above allow us to draw several conclusions that are applicable for the future developments of our trust model and its integration with social model and planning, as detailed in Section 7.3. In the first experiments, we have evaluated

6

Solving Inaccessibility in the Adversarial Environment

This chapter is dedicated to answering two principal questions – first part (Section 6.1) characterizes and analyzes the appropriateness of various techniques for solving inaccessibility, most notably extending the concept of stand-in agent, while the second part concentrates on the crucial issue that is common to all techniques described in the first part – efficiency. Efficiency is critical shall any solution succeed in real environment, as the robustness always comes with a cost and keeping this cost down (in terms of communication traffic as well as computational load) is a necessary prerequisite for the deployment of described techniques. Therefore, Section 6.2

6.1 Presentation of Inaccessibility Solutions

In the Chapter 3, we have presented the problem of inaccessibility in multi-agent system and introduced a method how to quantify and measure it.

We are now going to describe existing methods coping with inaccessibility. As defined briefly defined in Section 3.1.3, two main approaches can be distinguished between them: building *remote awareness* or *remote presence*.

6.1.1 Relay Agents

First, and perhaps the most classical solution to the inaccessibility problem are relay agents or low-level entities, responsible for setting up a transmission path through other elements when the direct contact between parties is impossible. Such protocols are currently widely implemented for routing in various types of networks, like TCP/IP or on lower levels [78]. However, this solution is efficient only if the network is in a "reasonably connected" state (see figure 3.1). Besides this limitation, that can be clearly distinguished in the results of our experiments, there are several other factors limiting the use of relayed connection. These factors are for example reduced battery life due to the fact that all the messages must be transmitted several times, or network maintenance overhead, especially in case of mobile networks. Another factor limiting the use of relaying in agent systems is the dynamic nature of their topology if the agent platforms are based on moving entities. In this case, relaying cost increases as the link maintenance and path-finding in dynamic environment is a non-trivial process.

6.1.2 Middle Agents

Middle agent is a term that can cover a whole range of different facilitators in a multi-agent system. In an overview article [74], author lists different types of middle agents - **Matchmakers** and **Brokers** (Facilitators). Matchmakers may provide remote awareness by notifying interested agents about the

presence of service providers, while the brokers can act as intermediaries and pass actual service requests between two mutually inaccessible parties. Even if this solution may perform very well in many situations, it may be unusable if middle agents are difficult to find, unreliable, or can not be trusted with private preferences of different parties.

6.1.3 Acquaintance Models

Social knowledge represent necessary and optional information which an agent needs for its efficient operation in the multi-agent community. The social knowledge is mainly used for reduction of communication, provides self-interested agents with a competitive advantage and allows agents to reason about the others in environments with partial accessibility.

The acquaintance model is a very specific knowledge structure containing agent's social knowledge. This knowledge structure is in a fact a computational model of agents' mutual awareness. It does not need to be precise and up-to-date. Agents may use different methods and techniques for maintenance and exploitation of the acquaintance model. There have been various acquaintance models studied and developed in the multi-agent community, eg. *tri-base acquaintance model* [50] and *twin-base acquaintance model* [14]. In principle, each acquaintance model is split into two parts: **self-knowledge** containing information about an agent itself and **social-knowledge** containing knowledge about other members of the multi-agent system.

While the former part of the model is maintained by the **social knowledge provider** (an owner), the latter is maintained by the **social knowledge requestor** (a client). There are two possible ways how the acquaintance model may be kept up-to-date, using *push* or *pull-mode* updates.

Social knowledge can be used for making operation of the multi-agent system more efficient. The acquaintance model is an important source of information that would have to be repeatedly communicated otherwise. Social knowledge and acquaintance models can be also used in the situations of agents' short term inaccessibility. However, the acquaintance models provides rather '*shallow*' knowledge, that does not represent a complicated dynamics of agent's decision making, future course of intentions, resource allocation or negotiation preferences. This type of information is needed for inter-agent coordination in situation with longer-term inaccessibility.

6.1.4 Stand-In Agent

An alternative option is to integrate the agent self-knowledge into a **stand-in agent** – a mobile computational entity that is constructed and maintained by the social knowledge provider [61]. While using stand-in, the social knowledge requestor does not create an acquaintance model of its own. Instead of communicating with the provider or middle agent, it interacts with its stand-in agent. Therefore, client agent is relieved from the relatively complex task of building and keeping up-to-date detailed acquaintance model and both provider and requestor may benefit from the full-fledged remote presence. Factoring the acquaintance model out of the each requestor agent internal memory allows it to be shared between all locally accessible agents, further minimizing the traffic and computational resources necessary for model maintenance.

Stand-in agents operates in two phases. During the **swarming** phase, stand-ins propagate through the system to reach the locations that may become inaccessible in the future.

In our implementation the community of stand-in agents operates in two phases: *stand-in swarming* and *information propagation and social knowledge synchronization*.

During the **swarming** phase, stand-ins propagate through the system to reach the locations that may become inaccessible in the future. First, existing stand-in agent or knowledge provider determines set of currently accessible locations using broadcast-like mechanism of underlying communication infrastructure. It analyzes the locations and decides which entities are interesting for further stand-in agent deployment, either because of the presence of knowledge requestor agent or because it considers the location to be interesting for future spread. Then, it may decide to create

and deploy its clones on one or more of these accessible locations. After its creation, each deployed stand-in agent chooses the type of functionality it will provide in its location and repeats evaluate/deploy process. The swarming propagation strategy is a crucial element of agent system tuning, as we must find a delicate balance between information spread efficiency and resources consumed by stand-ins.

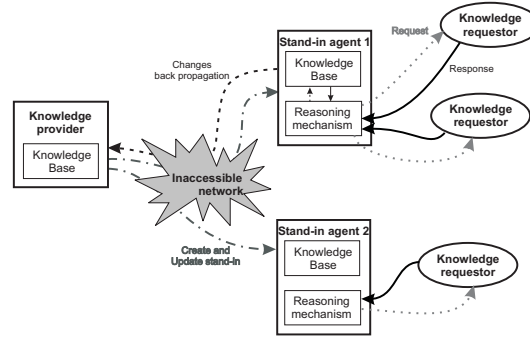


Fig. 6.1. The concept of the stand-in agent

Information propagation between members of the stand-in community is also a challenging process to tune. , because the information flows not only from the knowledge provider towards the stand-in community, but also from the stand-in community towards knowledge provider, or even within the isolated parts of the stand-in community. When a member of the stand-in community receives an update of the shared knowledge, or updates this knowledge after having acted on behalf of knowledge provider, it must determine if the information update is valuable enough to be propagated to other members of the community and eventually to the knowledge provider itself. It determines the list of currently accessible stand-ins in the community to which it will send the updated knowledge set or relevant subset and keeps the updated information ready for future synchronization with currently inaccessible stand-ins. In the presented original implementation, we use two different approaches to **information synchronization** phase. In the first implementation, we consider the cost of communication to be important and the stand-ins therefore synchronize their knowledge only when they encounter. When they receive an information update, they don't propagate it to other accessible members of the community. The second approach is based on an assumption that communication is cheap and that all updates are worth to be propagated to all accessible members of the community. They send every update to all accessible members of the community.

In this approach, any stand-in that updates the information or receives more recent version sends this update to all accessible members of the community. When two stand-ins become accessible, they exchange their information and join it into the shared common version, as ensured by domain-specific joining algorithm. This policy ensures an optimum information quality on domain elements, but must be optimized for domains with big number of locations and represented agents, for example using existing results from peer-to-peer networks research domain[24].

The most important added value of stand-in agent is not in providing remote awareness, but in providing rich, proactive and trusted remote presence by acting on behalf of knowledge provider. However, as in any system working on the shared data, synchronization problems arise in the agent community when the stand-ins accept commitments in place of knowledge provider.

6.1.5 Solution Selection

It is not always an easy task to pick a well adapted inaccessibility solution from those listed above. In the following paragraphs, we will use the accessibility metrics established in section 3.1 and determine which solutions are appropriate for measured operations and domains.

Domains can be easily classified in respect to their accessibility using the metrics provided above, but we shall also analyze the operations that are necessary to maintain the coordination between the members of the agent community. For each such operation we define two parameters: **the average operation duration** θ_ω and operation periodicity, the **mean time between operations**, π_ω .

Most protocols used to establish the remote awareness don't require complex interactions - one message, or one message with acknowledgments is sufficient for typical interaction (subscribe, inform). Therefore, the efficiency of the knowledge maintenance is determined mainly by the ratio between the parameter π_ω and $\tau_{\bar{g}}$, assuming that the accessibility time is sufficient for the transmission of a single message.

In the case of more complex operations, requiring remote presence and regular interaction with remote agents, both parameters θ_ω and π_ω are important. In the following, we suppose that both the $\tau_{\bar{g}}$ is bigger than θ_ω , as the case of frequent short-duration dropouts can be managed by network infrastructure. For an interaction to be successful, two conditions must be met:

- Interaction must start when the parties are accessible, requiring that the $\tau_{\bar{g}}$ is short enough in order to allow the interactions to take place sufficiently often, possibly in regular intervals.
- Interaction must be achieved before the connection is broken, requiring that $\tau_{\bar{g}}$ is bigger than θ_ω .

If the two conditions above can not be satisfied by direct interaction (implying direct/link accessibility) between two entities or using relaying (path accessibility), it is necessary to resolve the inaccessibility by using intermediary agents like brokers or stand-ins. Use of the intermediary agents redefines the problem and we found ourselves in the precedent situation, as we have replaced the communication between remote agents by the communication between agent and local (or consistently accessible) intermediary. This intermediary, implemented either as a broker or as a stand-in, is responsible for knowledge synchronization with the agent it represents using more sustainable protocols.

When the domain forces us to use intermediaries, we have to decide whether we want to use a stand-ins or brokers. As an input for this decision, at least the following questions shall be asked:

- Can the represented agent trust the intermediaries to follow its best interests?
- Are the goals, beliefs and strategies of the represented agent expressible in the language understood by agents and intermediaries?
- Can we benefit significantly from the synchronization between mutually accessible intermediaries, who are however inaccessible in respect to the represented agent?
- Are the platforms in our system strong enough to support a significant number of agents?

The four features mentioned in the questions can help us discriminate between the use of middle agents - brokers and stand-in agents. Stand-ins can not be in a situation with a conflict of interests, as they represent a single agent. Information exchange between the knowledge provider and stand-in agent is typically more efficient, because the dedicated agents can make more assumptions about the content of the information. It is also easier to represent the negotiation strategies in the stand-in, while their detailed disclosure to broker can be risky (Or even impossible if this agent lacks necessary abilities to execute them.) as these strategies may disclose agent's private information. Partial coordination between middle agents representing one agent in several locations can be much more difficult to implement, as it demands a deep knowledge about the structure of the representation data. On the other hand, the use of stand-in agent agents comes with a significant cost - we need the mesh covering the cooperator's locations to be sufficiently dense and we create one stand-in for each represented agent in many locations.

In the case of remote awareness, the situation is much simpler and the choice of mechanism depends on average accessibility between locations of collaborating agents, number of agents in each such location and the frequency of access to the social knowledge. This enables us to pick between relaying, matchmaking social knowledge or stand-ins.

6.1.6 Experiments

Experiments presented in this section establish the boundaries of applicability of the solutions to the inaccessibility problem presented in this section (Section 6.1). As we have already established the inaccessibility characteristics in the scenario, (see Section 3.2), we will now study how inaccessibility affects performance of our system. Three techniques for coping with inaccessibility will be analyzed, but we will show that the obtained results are applicable to all methods mentioned above.

For our measurements, we have prepared a simulation featuring a logistics problem in collaborative environment (the same scenario as abstractly described in the Section 3.2.1), where the humanitarian aid must be delivered to the zone ravaged by a disaster. In the domain, we will deploy three main types of entities: 5 aid sources, called **ports**, where the material comes in; 5 aid sinks, called **villages**, where it is consumed and 7 **transports** carrying the aid between ports and villages. Each transport has its predefined route that does not change during the simulation. Aid requests in the villages are generated by predefined script to ensure uniformity between simulation runs. They must be transmitted to the ports to ensure that the proper material is loaded on the transport going to the village. The way these requests are transmitted depends on the inaccessibility solution that is currently applied. We suppose that the physical communication links between the entities are limited-range radios, therefore the link exists if the distance is smaller then parameter ϱ . This parameter varies between different scenario runs to model different possible configurations, from complete link accessibility to only local (same position) accessibility.

In total, 33 results are presented, with 11 different communication ranges and 3 different approaches solving the inaccessibility problem:

- relaying transmissions by relay agents (6.1.1) – loading of the goods on a transport is possible only if a communication path exists between the destination village and the port in the moment when port-based entities negotiate the cargo to load,
- stand-in agents that only carry the information with no sharing in the stand-in community (see section 6.1.4),
- community stand-in agents, sharing the information updates with other members of the stand-in agent community (see also section 6.1.4).

To guarantee the uniformity of results, we have used the same negotiation protocols and workflow for the interaction between the acting agents and their environment. Both the requests in villages and goods in ports are generated from unique pseudorandom sequence used for all measurements. The only aspect that differentiates the scenarios is the mode of information transmission between requesting villages and goods providers in the ports.

Comparing the Solutions

After having determined the extent of inaccessibility in our system, we will study the effects inaccessibility has on the system performance. The system performance is given by a number of goods successfully delivered to villages. Zero value means complete failure, when no goods were transported, while 1 implies that all orders were completed.

On the following graph (see figure 6.2), we can observe the relationship between path accessibility and overall system performance for each of three measured solutions. Here we present the average requests coverage for different solution of inaccessibility. Results do follow the accessibility state partitioning from the previous paragraph. We can see that relay agents start to be reasonably useful

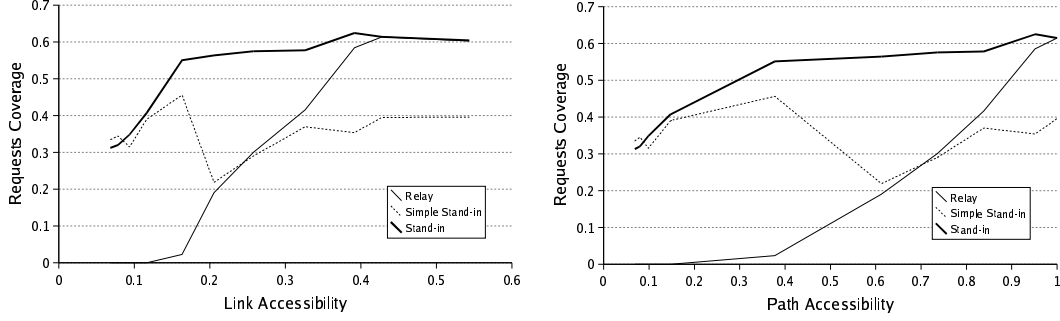


Fig. 6.2. The average requests coverage of three presented inaccessibility solutions and different accessibility settings.

when the link accessibility reaches 0.2, in the middle of the transition phase, well corresponding to the percolation threshold. Performance of isolated, non communicating stand-in remains constant. This is easy to understand, as these agents communicate only locally. They present an optimal solution for disconnected networks, as they require only a small number of messages to function.

On the other hand, performance of interacting community of stand-ins is more than a mere supremum of both previous methods. This is allowed by the dynamic nature of the system, where the stand-ins on mobile entities carry the up-to-date information through the system and spread it in small local communities, but relatively often. Thanks to this approach, the efficiency of system with these stand-ins approaches the optimum level with path accessibility of 0.4, instead of 0.9 for relay agents.

The goal of our measurements was to analyze the minimum and maximum performance of different solutions for inaccessibility. The measurements were carried out for three of the possible solutions described in section 6.1. We argue that the results we have obtained for stand-ins are the same for other solutions, middle agents and social knowledge, as the boundaries of performance of these solutions in a perfectly collaborative environment are the same. Note that we consider only the boundaries of performance and applicability of different methods. The synchronization load on the system is very different for each solution and the coordination protocols used by agents must be adapted to the selected approach and problem domain. Due to the fact that most operations in our system were very short in duration, the conclusive measurements of impact of other inaccessibility parameters, like τ_{θ} and $\tau_{\bar{\theta}}$ were impossible to obtain.

Evaluation of Experiments

As we have shown above, stand-ins and broker agents provide more than a viable alternative to message relaying in environments with low link accessibility or high cost of communication. They allow efficient coordination and collaboration in communities with low and transient accessibility and they match the performance of relaying in connected communities. However, the implementation of the intermediary agents for a given domain is not trivial and their use in larger communities of agents requires some additional tuning of two principal methods they use – swarming of the stand-in agents and knowledge distribution/synchronization between intermediaries.

In the next section, we will address the most important problem related to the introduction of above-mentioned techniques to the multi-agent system. We have seen that around and above the percolation threshold, the number of messages used for synchronization increases dramatically and that the number of stand-ins or other middle agents can locally or globally exceed the necessary number. Therefore, as we present in Section 6.2, we have focused on efficiency concerns and we have devised a method that may be combined with all techniques defined in this section.

6.2 Optimizing the Inaccessibility Solutions

In Chapter 3, we have provided a definition of inaccessibility and the mathematical model that correctly describes the inaccessibility in a geographically distributed multi-agent system. In the previous section, we have also investigated several possible solutions for inaccessibility and evaluated their appropriateness in different types of environment - from nearly inaccessible to almost completely accessible.

In this chapter, we further extend the concept. Following the results from the experiments, we are now trying to optimize the behavior of the multi-agent system to make it more efficient (by reducing the number of messages), while maintaining the robustness with respect to inaccessibility.

In dynamic systems, possibly with mobile agents, the accessibility values change over time. In those case, time-averages are used to describe the system nature.

Different values of either of the accessibility quantities specify completely different set of problems to be solved. In the situations with *low path accessibility* we need to investigate mechanisms how the agents can coordinate and plan commitments even if they are temporarily inaccessible. Providing solutions for these problems is highly domain specific. In the situations with high path accessibility and *low link accessibility* we need to optimize the forwarding mechanism so that the optimal number (and right location) of the relay nodes is used. The second class of problems, which is substantially more domain independent, will be discussed herein.

Inaccessibility solutions, as introduced in the previous section, can be divided into mechanisms providing **remote awareness** – providing the inaccessible agent with the information about the inaccessible part of the infrastructure and **remote presence** – an ability of inaccessible agent to act remotely (by e.g. stand-in agents acting on the inaccessible agent behalf [61]). Some inaccessibility solutions are based on agents individual maintenance of their social knowledge and acquaintance models [42] or by deployment of various kinds of **middle agents** such *matchmaker agents* [73], *brokers* [74] or *stand-in agents*. A generic model of the middle agent, based on the stand-in concept, is suggested and presented in Section 6.2.1. Throughout this Section, term middle agent will be used as generic denomination for matchmakers, brokers and stand-ins, as our findings are applicable to all above mentioned methods. However, all the measurements and experiments were done using stand-in agents.

In this deliverable, we address the problem of optimal distribution of middle agents in the distributed dynamic multi-agent system and message flow optimization between these agents to ensure optimum balance between efficiency and redundancy. We don't explicitly restrict the algorithm use to particular type of middle agent, as it can be integrated with all the technologies defined above.

We shall note that in the real dynamic systems, the accessibility values are rarely homogenous. Clusters of agents often tend to move together and encounter other agents directly or via relays. Therefore, the optimization algorithm must (i) ensure the existence of middle agents and message transmissions to *optimally connect*¹ the multi-agent system, while (ii) being *local* in the sense that middle agents shall be able to operate with the local environment information only, without the need for the central coordination. *System adaptivity* (iii) is another crucial factor, as it shall autonomously adapt to current local environment both in time and in place, but the whole algorithm shall be also (iv) *efficient* in the number of messages consumed and computational load. *Community stability* (v) is closely related to efficiency as the overly rapid adaptation can significantly increase computational load of the system.

The algorithm we propose is based on virtual payments combined with social dominance [75, 65] model, as detailed in Section 6.2.2.

¹ Optimal connectivity is defined as a ratio of mutually accessible agent pairs and a proportion of the messages actually delivered.

6.2.1 Middle Agent Architecture

In this section we will address integration of the algorithm (see Section 6.2.2) with the generic middle agent architecture. Unlike classical middle-agent architectures [73] where the prime functionality is devoted towards matchmaking and negotiation, we would like to extend the concept of middle agent by its capability to autonomously migrate in the network, clone and destruct copies.

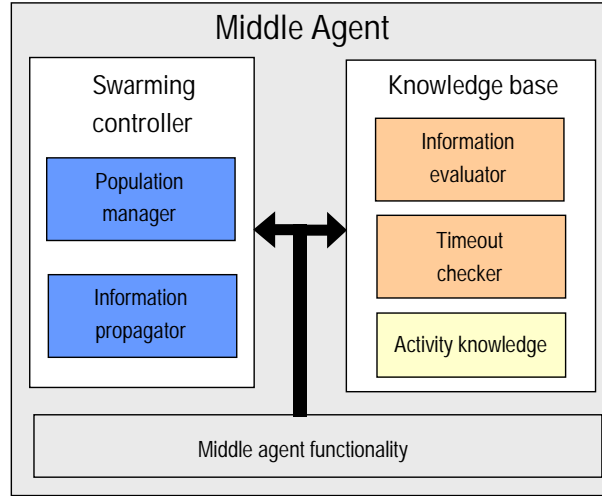


Fig. 6.3. Middle agent architecture.

In Figure 6.3 we present our algorithm component – *swarming controller* in the context of middle agent architecture.

- **Swarming controller** – consists of two modules: **population manager** ensures cloning, migration and destruction of middle agents in the system while the **information propagator** manages information flows through the agent, more specifically the messages or knowledge to transfer or actions to take. The module must balance between two extreme cases of knowledge handling: propagation to all visible targets or no propagation at all. Even if both modules are domain independent, they depend on the domain specific functions included in the knowledge base algorithms. Details of the algorithm used are described in the dedicated section 6.2.2.
- **Knowledge base**, a domain specific knowledge structure of the middle agent, consists of three parts: **activity knowledge**, *information evaluator* and *timeout checker*. While the **activity knowledge** contains the domain specific knowledge and the meta-data provided by the propagator, the information evaluator and timeout checker are the algorithms working on this knowledge. The **information evaluator** classifies and indexes the knowledge, so that the index values can be used by information propagator to manage its activity and further propagation. It also evaluates the knowledge usefulness. The **timeout checker** module implements *forgetting* of the activity knowledge.
- **Middle agent functionality** – universal interface between modules and agent platform. It provides fundamental agent functions (clone, migrate and die), message interface and monitoring listeners, as well as original middle agent code. This code depends on the actual type of the middle agent. Via monitoring listeners it notifies modules about visibility of the other nodes, information about accessible other middle agents and also about presence of potential message receiver. Only this part of relay agent needs to be changed to work properly with another agent platform.



Fig. 6.4. Middle agent network covers the ACROSS domain.

Besides middle agents, the later presented algorithm can be also integrated with simple relaying agents whose functionality is based on sole, uninformed message forwarding.

On the Figure 6.4, we present a sample middle agent network in ACROSS domain, as described in dedicated part of the document: nodes represent fixed locations, lines between nodes means that there exists a communication link between them and residing middle agent is represented by point near the node. In the next section we will discuss the details of swarming controller module that has placed the middle agents in the system.

6.2.2 Swarming Controller

As mentioned above one of the key issues in middle-agent operation is their proper location in the network. The distributed middle-agent allocation mechanism uses only locally accessible information. It does so not only minimize the network maintenance communication, but also allows operation in the disruptive or partially inaccessible environment. Locally accessible information is obtained by monitoring middle-agents neighborhood – identifying currently visible targets and other middle agents. The algorithm needs to be lightweight and computationally simple, as the middle-agent instances can be constrained by the devices they run on. Scalability in space and density shall also be an important property of the targeted solution.

In principle there are two key approaches to controlling the efficiency of the middle agents allocation:

1. *forward swarming control* – where the middle agent migrates its clone only to the locations with higher possibility of future inaccessibility and higher interaction expectancy and
2. *backward swarming control* – where the middle agents dispatch their clones to every reachable destination and the useless ones are eliminated in the future following the inaccessibility real situation.

Each of the approaches has its pros and cons. The forward swarming control is computationally efficient, as it tries to minimize the number of stand-in agents in the system and prevent the possible swarming explosion. This is why that approach seems to be particularly suitable for domains with high scalability and operational efficiency requirements. On the other hand, the backward swarming control has got an important advantage. This approach is substantially more domain independent, demands less knowledge about the environment nature and is more robust, as it doesn't *explicitly* use any prediction about the future of the community.

In this stage of the project, we have opted for the use of the *backward swarming*, as this approach is more robust and domain independent.

Abstract criteria of the system quality defined in the introduction were also formulated in a precise manner, with descending priority:

- Provide connection between any two system elements through the minimum number of middle agents.
- Minimize the number of middle agents in the system.
- Minimize the number of messages for system operation and/or knowledge maintenance.

Population manager is driven by a biology inspired algorithm. Social dominance and altruism models [75, 65] were successfully used to partition the group of agents into those who work for the good of the community and the others, who profit from the altruism of the first group. During the experiments with rats, it was determined that a exactly the sufficient number of individuals behaves in an altruistic manner to optimize the *whole group* fitness. They bring the food and share it with the others, who only consume. This behavior is formalized by a simple mathematical model formulated in [75].

To ensure the target coverage, middle agents can be reproduced in the system using two main propagation strategies:

- *full flood fill* – any middle agent initiates full flood filling reproduction strategy when it identifies a new unserved knowledge target in its reach. To decide whether the target is really new, all agents keep the set of served targets, that includes both the other middle agents and knowledge final users. Target is removed from the set when it is not used for specified duration – *forget time*. In practice, the middle agent is cloned to every visible node where it is not running yet if the new knowledge target is not reachable from current position of the agent. Created clones further clone themselves to new locations without existing middle agents using the same cloning termination condition: target reachability. Only this simple strategy can ensure that the middle agent network will reach the target. Using random walk instead of flood fill is possible, but not advisable, because the random walk does not guarantee finding the target, as known from Pólya's random walk theorem² [3],
- *bounded flood fill* – this is depth-limited version of the previous reproduction strategy. After the initiation, the middle agents are successively cloned only to depth specified by **FloodFillDepth** constant. This reproduction strategy is triggered by a local accessibility change when the source agent holds relevant, non-expired knowledge. Application of this mechanism can identify shorter paths enabled by the accessibility change or can deliver the knowledge to the isolated cluster by the middle agent on the mobile node.

Both flooding strategies are time limited. There is specified constant *flood duration* during which the middle agent retains reproduction intention. When this period expires, the agent no longer reproduces until the new reproduction is started by the agent itself or the others.

To keep the number of middle agents close to optimum, the population manager contains also the methods that decrease the number of middle agents in the system:

- In *random duels* the attacking middle agent randomly selects an adversary between accessible agents and launches an attack with force proportional to its profit during specific period, as determined by *information propagator* (see below). Besides the attacks force, the attack also

² Pólya considered a d -dimensional array of lattice points where a point moves to any of its neighbors with equal probability. He asked whether given an arbitrary point A in the lattice, a point executing a random walk starting from the origin would reach A with probability 1. Pólya's surprising answer was that it would for $d = 1$ and for $d = 2$, but it would not for $d \geq 3$. In later work he also analyzed two points executing independent random walks and also at random walks satisfying the condition that the moving point never passed through the same lattice point twice.

includes the information about its *active target* set at the time of the attack. The attacked agent evaluates the attack and decides whether it won or lost. If the attacked agent loses, it removes itself from the system; losing attacker is not penalized for the attack. Attack evaluation compares the *active targets* first and when one is a subset of the other, its owner loses the fight. When the sets are identical, force of the attack decides the fight – stronger agent wins. Active target set size is evaluated differently for new and old agents. For the young agents that are not yet completely adapted to the environment, the set contains all directly accessible targets, while it contains only the really used ones. Besides this advantage, the youngest agents benefit from the immunity period, during which they can not lose a fight while attacked.

- In contrast to the previous case, the *uselessness detection* is an individual process. The middle agent can remove itself while it is isolated from the rest of the community and no access to relevant knowledge anymore.

Information propagator manages knowledge propagation and use in the system. This component uses virtual payments to reward the other agents for the knowledge, receives payments from the others for the information provided and generates the profit also from acting on behalf of the represented agent. Each agent optimizes its profit, ensuring the overall information flow efficiency. More specifically, middle agents reward the information received from the others in function of information usefulness and redundancy – the first agent from which they receive the information receives significant payment, while the subsequent information is rewarded less. On the other hand, the transmitting the information to other agents is not free of charge for the agent – it must carefully decide to which agents it propagates which information. The decision is taken in function of the previous experience (and current network status) with similar knowledge, and the knowledge source and potential target are important, domain independent similarity criteria. Besides these criteria, we may enhance the knowledge with meta-data specifying to which agent it has been already provided. To make the system more robust, the decision to which nodes we send the information is probabilistic – agents may therefore send the knowledge to the less rated directions to find better paths in the system. Payment for the information is transmitted as a reply to the knowledge update message.

Historical data (represented as probabilities assigned to knowledge characteristics, origins and targets) that are used to identify the targets to which we send the information are periodically updated and the old data is discarded. When a new target appears, the initial message transmission probability is set to the level derived automatically from the current network state of the evaluating agent.

The domain specific functions that evaluate the usefulness of the knowledge, indexable knowledge characteristics and rewards for actions are provided by the domain-specific knowledge base.

We shall keep in mind that the knowledge is not only propagated by messaging in the network of middle agents, but also carried by the agents created during the reproduction process. This is especially important for the communities where the accessibility is relatively low, or where the agents are clustered.

6.2.3 Experiments

In this section, we describe a set of experiments with middle agents in fixed and partially mobile mobile ad-hoc network. Agents are running on simulated nodes and can migrate between accessible nodes. In the system, both fixed and mobile nodes are used.

A pair of containers is mutually accessible if each of the two containers is located within the visibility range of the other container. In the experiments, we use testing agents that implement a FIPA CNP [67] in which all requests must involve at least one middle agent, even if a direct link between the sender and the receiver exists, and we didn't use any advanced middle agent capabilities to keep the results middle agent type-independent. Therefore, our agents worked in a simple relay mode, not using any social knowledge. However, the use social knowledge typically further decreases the number of messages necessary [53].

When system is started, no middle agents exist. The first middle agent is created by the sender who wants middle agents to deliver its request to one or more targets. This first middle agent then propagates using the full flood fill as described in Section 6.2.2.

6.2.4 Information Propagator Adaptation

In the first experiment, we have deactivated the *population manager* to keep one middle agent and one test agent on each of 24 nodes in our network. As a "worst" case scenario, full link accessibility case was set to obtain the slowest convergence of the number of messages towards the optimum, as many loops and alternative paths exist in the network. In the scenario, one of the testing agents periodically starts a testing CNP to all test agents. The results (Fig. 6.5) show the systemwide number of transmitted messages per each CNP round. We provide the results for two different values of the forgetting parameter. This parameter should be zero in cases of non-changing topologies because all knowledge previously stored in information propagator can be reused and the system rapidly converges to the optimal number of 119 messages per CNP round. When we increase the value of the forgetting parameter, the number of messages per round can't converge to the optimum because a certain amount of the knowledge is being lost. The experiment shows that the number of messages is decreasing exponentially until a certain threshold is reached. After reaching the threshold, the number of messages per round remains more or less constant. The experiment also verified that in case of the full accessibility, any routing path contains exactly one middle agent.

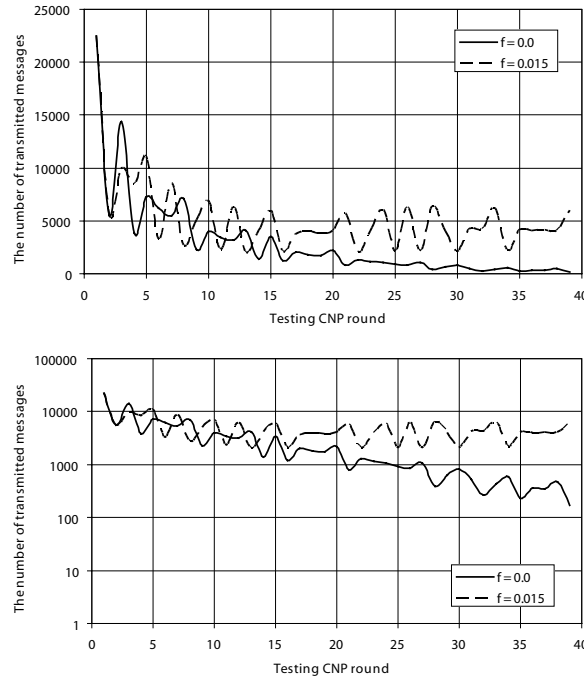


Fig. 6.5. Message reduction using different forget parameter (f) value in fully connected environment with 24 nodes. Bottom chart has logarithmic scale on Y-axis.

6.2.5 Adaptation to the Changing Environment

In this experiment, the visibility range was set to minimize the link accessibility, but to maintain complete path accessibility. Testing agents and relay agents were set up as in the previous experiment, but the population manager was enabled this time, causing the number of relay agents to vary over time. We measured the evolution of the number of middle agents at three levels of inaccessibility dynamics. To determine the optimal number of stand-in agents in each moment, we have implemented an efficient centralized algorithm briefly presented in the next paragraph. In our domain, where the accessibility is distance-based, this algorithm behaves optimally.

Reference Solution

This algorithm finds the shortest routing paths between all pairs of the agents and selects a subset of these paths that is supported by the minimal number of middle agents.

The communication environment is described as a non-oriented and non-valued graph $G = (V, E)$ where V is a set of nodes and E is a set of edges between link-accessible nodes.

Let set **solved** contain the nodes where the middle agents should be placed. **distance**(c_i) is a number of edges in a shortest path p_i between the couple of nodes c_i . The algorithm processes all couples c_i with **distance**(c_i) = 2. Let $\mathbf{fp}_{c_i} = \{p_i\}$ is a set of the shortest paths between a couple of nodes c_i . **mid**(p_i) is the node in the middle of a path p_i , where **length**(p_i) = 2.

```

solved =  $\emptyset$ 
nearCouples =  $\emptyset$ 
C = set of all couples of nodes
for ( $\forall c_i \in C$ ) {
    if (distance( $c_i$ ) == 2)
        nearCouples = nearCouples  $\cup$   $c_i$ 
}
while (nearCouples !=  $\emptyset$ ) {
    allConflicts =  $\emptyset$ 
    for ( $\forall nn_i \in \text{nearCouples}$ ) {
        if ( $\forall p_i \in \mathbf{fp}_{nn_i} : \mathbf{mid}(p_i) \notin \text{solved}$ ) {
            conflicts =  $\emptyset$ 
            for ( $\forall p_i \in \mathbf{fp}_{nn_i}$ )
                conflicts = conflicts  $\cup$  mid( $p_i$ )
            }
            allConflicts = allConflicts  $\cup$  {conflicts}
        }
    }
    addThisRelay = most frequented node in allConflicts
    for ( $\forall nn_i \in \text{nearCouples}$ ) {
        if (mid( $nn_i$ ) == addThisRelay)
            remove  $nn_i$  from nearCouples
    }
    solved = solved  $\cup$  addThisRelay
}

```

The algorithm returns **solved**, the minimum set of nodes where the middle agents must be placed to connect all couples c_i with **distance**(c_i) = 2. Using induction, it can be proved that the **solved** set supports also the shortest paths between any couple c_i in the system, regardless of their distance.

The algorithm takes a couple of nodes c_i whose **distance**(c_i) = 2 and determines all nodes that lies on the fastest paths between the couple. If there is no middle agent on either of these nodes they are placed into **conflicts**. In **allConflicts** are separately stored **conflicts** generated by all the couples in the graph. The most frequently added node in **allConflicts** is added to **solved** set. After that the **allConflict** is set clear and the algorithm repeats until all two-edge paths are interconnected by a middle agent.

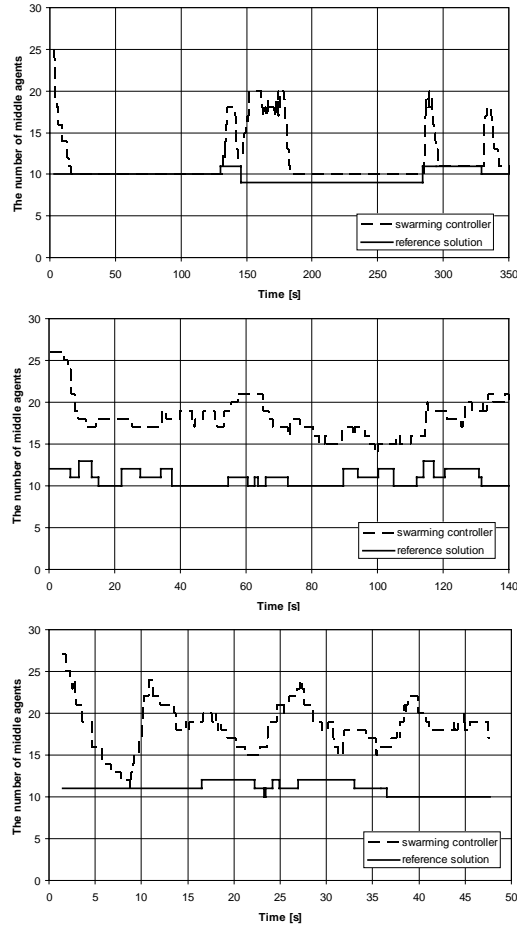


Fig. 6.6. The figure presents adaptation of middle agent network to the changing environment with different speed of changes: from infrequent changes (top chart) to the fast, frequent changes (bottom chart).

Results

At first (Fig. 6.6, top) we were slowly changing visibility ranges on the network with fixed nodes. In the second setup (Fig. 6.6, middle), we have added 1 mobile node into the network. The movement of the mobile node through the whole network introduces local accessibility changes. In the third setup (Fig. 6.6, bottom), two mobile nodes were moving faster through the community, causing more important disturbances in the network topology.

In the experiments, we clearly demonstrate that the agents are able to organize themselves efficiently and to approach the optimal number as determined by the reference algorithm. However, after the steep initial decrease, we can observe the peaks that correspond to agent propagation in response to the topology changes. In the mobile scenarios, we have failed to match the reference solution perfectly, as the adaptation time is somewhat higher than the average change period, but the results remain comparable and the robustness and distribution of the algorithm provides enough of the incentive for its application.

6.2.6 Observations

In the experiments, we show that our solution is efficient in communication, is robust with respect to important environment changes and ensures complete system connectivity in the environments with high path accessibility and low link accessibility. The mechanism can be integrated with any type of middle agent and its compatibility with various middle agent types makes it applicable also in disconnected environments. So far, our experiments have proved that mechanism performs well in the environments with high path accessibility and relatively small number of mobile entities or accessibility changes. In the following chapters, we will present the integration of the mechanism with stand-in agents to efficiently support coalition cooperation in inaccessible, adversarial environment.

7

Efficient Teamwork in Inaccessible and Adversarial Environment

This chapter presents the design of a prototype solution integrating the concepts developed in this project into a coherent system solving the Humanitarian assistance problem presented in Section 4.4. While the problem (and part of the implementation) are certainly domain dependent, most of the techniques are generic - they can be used to solve similar logistical or other distributed coordination problems, by carefully integrating them into the system appropriate for the problem environment that can be characterized using the techniques from Chapters 2 and 3.

While the Section 7.1 describes the problem solved in a generic manner, Section 7.2 describes the techniques used to solve the inaccessibility related issues, and Section 7.3 presents the integration of trust model with social model and efficient distributed planning techniques that take into account the disclosure of private information and trust data from the model.

7.1 Problem Statement

On the background of the scenario presented in Section 4.4, the Humanitarian Agents must lead the system to solve following tasks:

1. Acquire the knowledge about the disaster extent and the needs of the population in the disaster area.
2. Using the information from the previous step, plan the quantities of the goods to deliver to various locations in the disaster area.
3. Humanitarian Agents have no transport of their own. Therefore, they must use the services of other Transporter agents to actually deliver the cargo. Humanitarian agents are keep the role of the coalition leader, therefore, they must be able to plan the transport efficiently, taking into considerations the following aspects:

Resource Availability for all potential members must be taken into account to avoid unnecessary negotiation and to use the available resources in an efficient manner¹. Therefore, agents must maintain a model of the other's agent resources in order to be able to plan locally, but with relevant information and resort to negotiation only in later stages of the planning process.

Private Preferences of the coalition members are never openly communicated to Humanitarian agents or other agents. Therefore, each agent must agree to participate in a team each time it is composed, requiring a multi-step negotiation.

Private Information Disclosure must be minimized. Therefore, transporter agents that act as coalition members don't typically disclose the complete information about their resources,

¹ In such a complex problem, claiming optimality is nearly impossible.

but only the aggregate information that is sufficient for high-level planning, but won't endanger the agent's assets.

Adversariality Consideration : the solution we propose shall be robust with respect to adversarial actions. It shall be able to estimate the effects of adversarial actions on the plan execution and to adapt the plan accordingly, as well as consider the trustworthiness of coalition members while assigning the tasks. Therefore, accumulating the previous experience in appropriate form (trust model in our case) is necessary. As trustfulness data are only rarely exact, the planning must be robust with respect to errors, uncertainty or bias in the data.

4. Even if most operations happen outside of the accessible/observable area, coalition leader must be able to gather data on individual actions results and other events during plan execution. This data can be used for two purposes: current plan replanning and update of the trust values for future planning.

To solve the above issues, we have designed the system depicted in Figure 7.1².

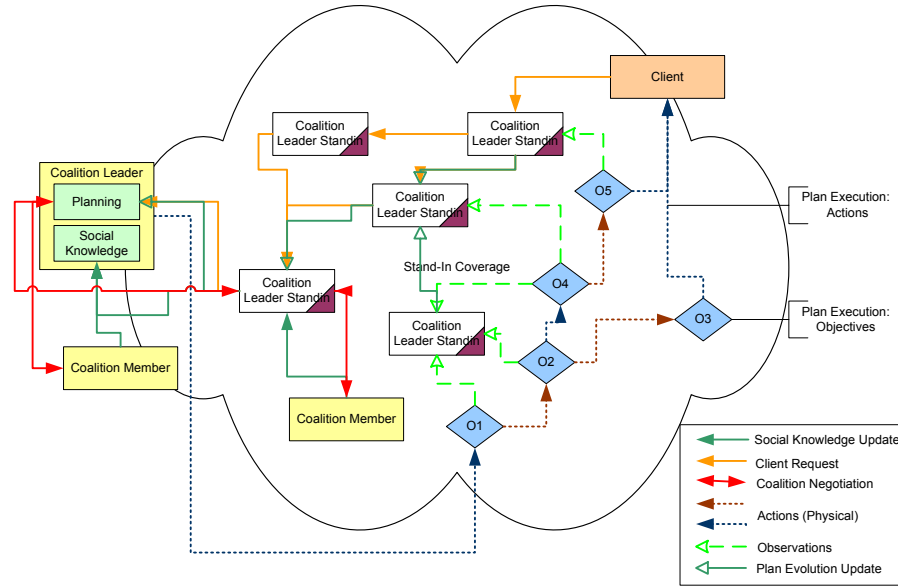


Fig. 7.1. Prototype Overview: Coalition cooperation in inaccessible and adversarial environment.

In the Figure 7.1, we can distinguish the dashed lines that represent physical actions between agents (or rather their respective hosting devices), while the solid lines represent the communication acts. At first, the Client Agent (Location agent in a disaster area) demands the help in form of goods delivery from the humanitarian agent stand-in deployed in the area. Stand-in updates its knowledge with the request and in turn propagates the information through the stand-in network to its owner (orange lines). Humanitarian agent will then assume the role of coalition leader and will use the social knowledge it has about transporter agents (green lines represent social knowledge maintenance) to prepare preliminary action plan that is negotiated with transporters and finally

² To simplify the development, we have parted with an assumption that all Transporter Agents who participate in the coalition are accessible to coalition leader (Humanitarian Agent). This assumption doesn't endanger the generality of the approach, but saves us from tedious and costly re-implementation of all protocols used by Transporters through their stand-in agents. On the other hand, the communication between Location agents in the disaster area and Humanitarian agent passes through stand-in network.

launched (red lines) as detailed in Section 7.3. As soon as the plan execution is launched, coalition leader’s stand-ins in the disaster area receive the task description and subscribe with Location agents in their vicinity to observe fulfillment of objectives (deliveries or failures to deliver), represented by light-green dashed lines. The observed data is communicated to the owner (green lines), also through the stand-in network mechanisms.

In the following paragraphs, we will present the elements of our solution that were introduced in the overview and discuss the techniques researched in the course of the project as they were integrated.

7.2 Stand-In Agents

The adversariality of the environment makes the choice of the inaccessibility solution technique as restrained process – middle agent or relays are potentially adversarial and can not be trusted with sensitive data. Therefore, the solution is a combination of social knowledge (see Section 6.1.3) with stand-in agents (see Section 6.1.4), where stand-in agents gather the social knowledge and take commitments on behalf of the humanitarian agent. Commitments and social knowledge then serve as an input for the planning. Swarming of the stand-ins of the humanitarian agent is typically complete and unrestricted due to the highly dynamic nature of the network introduced by the vehicles. In our experiment, we assume that any container is ready to host humanitarian agent stand-in and that the agents it encounters provide it with necessary information and services upon request.

Humanitarian stand-in directly communicate with location agents in disaster area and participates in the auctions – this is allowed by the local presence as auction requires multi-stage interaction. It makes a proposal independently, and if accepted, communicates the data to the Humanitarian Agent that organizes the transport and transmits back the actual coverage. This transmission is not that time critical and both the request and answer pass through the stand-in network.

Once the coalition is formed and transporters have committed to their tasks, the stand-ins receive the complete information about the task assignment, as shown in Figure 7.2. They analyze the information, select the actions that are relevant to them and use SUBSCRIBE protocol to receive the information about deliveries and failures from the accessible locations. Once they make the observation, they update their knowledge and use the information propagator to select the set of stand-ins to convey the information to. The same operation is performed when the observation is received from another stand-in. Therefore, the Humanitarian agent is informed about the results of the plan execution and can act accordingly.

As a general design principle, the social model has to be shared pragmatically, as sharing induces synchronization costs and can cause security risks if the security of the host platform of the stand-in is breached. From the above description, we can see that the stand-in doesn’t need to maintain the complete copy of the social knowledge of the owner agent: we restrict its knowledge to the following elements:

- **Location demands**, the stand-in has committed to, are transmitted to owner as an input for the planning process.
- **Plan Breakdown and Task Attribution** to team members for each coalition the owner agent leads: this data is used by stand-in to gather the information about task accomplishment that is not observable by its owner due to the inaccessibility. This knowledge structure also stores the task results observed or received by the agent.

When the knowledge is no longer necessary, it is discarded automatically and the corresponding SUBSCRIBES are terminated as well.

The tricky part of the knowledge maintenance to handle in this scenario are multiple observations of the same event. In our case, the original event message ID³ is included in the update and allows

³ Assumed to be unique.

the agent to check for duplicity. If such ID is not available, similar checks must be done using the content of the message.

Information propagator depends on probabilistic routing model. This model contains probabilities of transferring the knowledge to other accessible stand-ins. The model maintains its efficiency by incorporating the virtual payment system - each message any stand-in receives is rewarded according to its significance and novelty: in our domain, demands from locations are more significant than observations of batch deliveries. Novelty is assessed on a more complex scale, where the first agent, who communicates the knowledge, receives a high reward, the second one gets significantly less and the subsequent payments are rapidly decreasing under the cost of message sending.

Propagation mechanism was improved by the introduction of "tombstone". When a stand-in agent dies on a container, it leaves behind a simple sign ("tombstone") to prevent the cloning of another stand-in from different source. The grave disappears automatically after predefined period of time, allowing readaptation in case of environmental change. This feature is important in highly variable mobile environments, where the creation/destruction rate would be too high otherwise.

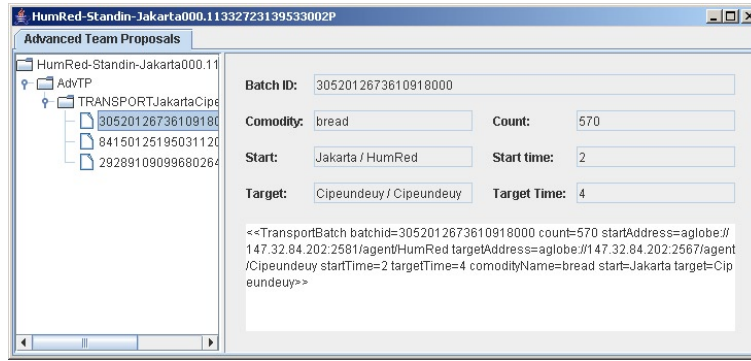


Fig. 7.2. Humanitarian stand-in GUI featuring one surveyed plan decomposed into individual tasks and batches.

7.3 Trust-Based Planning for Adversarial Domains

In this section, we address the problem of collaborative logistics planning in uncertain, self-interested and adversarial environments, while not considering the inaccessibility for the moment to simplify the reasoning. This problem is significantly different from classical cooperative planning due to the requirements listed below:

- **limiting information disclosure** to other agents, respecting each agent's private preferences, and keeping them undisclosed;
- **integration of trust model** [16] and methods of reasoning about competitive and adversarial agents,
- **stability** - we want the solution (plan) to be stable even if a trustworthiness or availability of partner agents changes slightly;
- **efficiency** - we want to be able to find a task decomposition and provisional allocation within reasonable time and with a relatively small number of messages.

At first, we need to update the trust model presented in Chapter 5 to better integrate it with social model and prepare it to be used as a valuable input for planning algorithm. Figure 7.3 presents the differences between use of the model as presented above – trusting decision is taken on each agent independently on the context, using only the self-trust as a parameter. In this figure, we can see that

the trust model is tightly integrated with social model elements that are used as a primary planning input, furthermore, most of the restrictions in the planning model that follows use trust in agents or actions as an input to ensure reasonable team selection and adaptation to local (action dependent) trust levels.

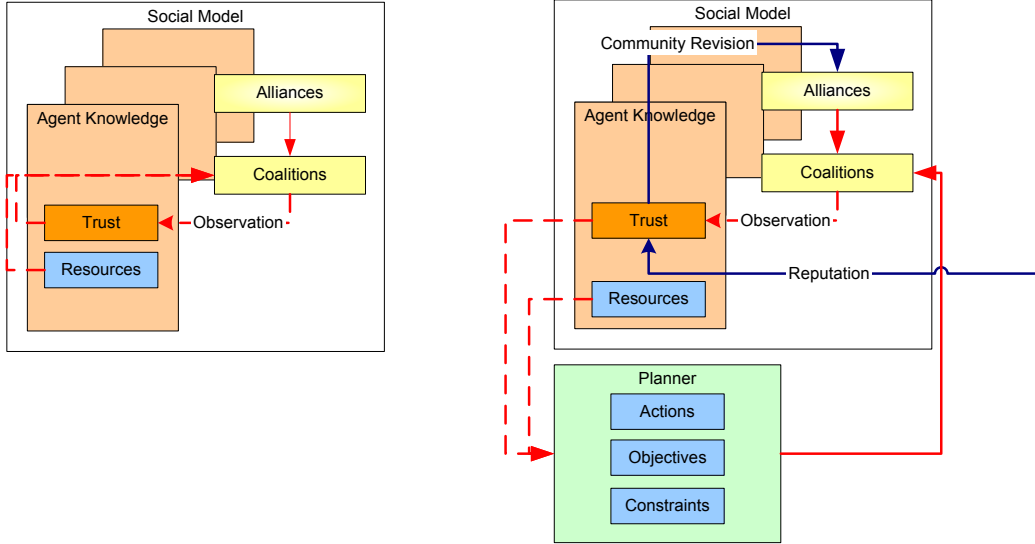


Fig. 7.3. Evolution of the model from the original, simple version, towards a more complex, better integrated version.

The algorithm we present can be compared to abundant previous works in the MAS logistics area [21]. Enhanced Contract-Net-Protocol as defined in [28] and further extended towards practical application by [55] achieves the same result using the negotiation between coalition leader and perspective members. When the perspective coalition leader wishes to solve the task, it asks other agents to cover the task completely or at least partially. Agents submit their bids, the best ones are selected and provisionally granted the task. The rest of the task is auctioned again and new auctions are organized until the whole task is covered. If the remaining task can not be covered, the algorithm must achieve consistency by backtracking – revocations of provisionally granted tasks and auctioning new ones. Even if we have a unique coalition leader, the planning problem is completely decentralized and requires intensive communication. Consequently, this approach presents performance problems when it prepares the initial plan in large state spaces – even if such planner compares favorably with humans [56, 36], it can be easily beaten by mathematical programming techniques if we are able to formulate the problem appropriately.

On the other hand, the agent approach brings more flexibility than mathematical programming as the agents may combine many sources and types of knowledge to prepare the plan, each agent contributing its knowledge, reasoning and resources. Agents do not need to be aware of each other's mental states, provided that they are syntactically and semantically interoperable.

Presented approach here is an attempt to integrate classical artificial intelligence and operational research 'heavy-duty' solvers in the contest of multi-agent systems. We argue that the abstract models of collaboration in agent systems as they are now used within the multi-agent system community have severe drawbacks – they are well suited for simple reasoning and limited amount of knowledge, while little scalable. Their performance tends to degrade with increasing problem complexity and shift the focus from qualitative to quantitative reasoning. Therefore, we propose that the AI/OR techniques are a very good fit for agent reasoning due to their high performance and little or no scalability

problems. The traditional problems related to their application – restrictive applicability conditions (e.g. linearity) are solved by modern methods [15] and on the other side, acquaintance models [52] provide the necessary knowledge inputs for the model, as well as an efficient mechanisms for its maintenance. As the mechanism we propose is intended to function in adversarial environments, we need to augment the social model with trustfulness information, using one of the available trust and reputation models: [62, 16, 63, 37] and others [60].

Such trust mechanism must comply with following requirements: (i) trust and reputation must be tightly integrated with the planning mechanism, (ii) model must be robust with respect to environmental noise (natural failure), (iii) the model inputs must be compatible with the observed plan outcome.

The following section provides the formal statement of the planning problem we are trying to address, in the Section 7.5 we describe the planning algorithm, a key contribution of this phase of work. As mentioned earlier, this algorithm consists of local planning algorithm (see Section 7.5.1), local plan evaluation (see Section 7.5.2), coherence and verification (see Section 7.5.3) followed by plan execution. In Section 7.5.4 we discuss some properties of the designed algorithm.

7.4 Formal Problem Statement

In the logistics planning problem we consider, we address the transport of goods from initial to terminal location using the resources belonging to self-interested and potentially adversarial agents. Therefore, we must select appropriate routes from the plan base, combine them and allocate resources to the tasks in the plan in order to maximize the expected amount of delivered goods. In the formal problem presentation below, we present the problem from the perspective of the coalition leader – the agent denoted A_0 that needs the cargo to be transported and organizes a coalition to carry it.

Formally, we follow the approach proposed by [77] and instead of decomposing the plan into the action-state graph, we will describe it using actions and objectives (called objects in [77]). Therefore, we will define an **abstract plan** (e.g. route plan) as a directed bipartite graph, where one side is composed of **objectives** (typically corresponding to locations), defined by the set $O = \{o_0(initial), o_1, o_n(terminal)\}$, with each member defined as $o_i = (prer_{o_i}, allows_{o_i})$, where both the $prer_{o_i}$ and $allows_{o_i}$ are subsets from the Ac , while the other graph side contains **actions** (transports) linking the objectives, defined in the set $Ac = \{a_1, a_2, \dots, a_m\}$, where again $a_i = (prer_{a_i}, allows_{a_i})$ and sets $prer_{a_i}$ and $allows_{a_i}$ are subsets of O . By definition, we always start from a single *initial objective* o_0 (with no prerequisites: $prer_{o_0} = \emptyset$) and terminate in a *terminal objective* that corresponds to the achieved goal state: $allows_{o_n} = \emptyset$. This formal simplification doesn't reduce the generality of our approach - in case of need, we may define formal zero-cost actions from/to the initial/terminal objective.⁴ Besides the structural information, we also keep Θ_{a_i} for each action – an estimate of normal action trustfulness as obtained from the trust model.

Batches constitute the cargo that is transported. For the purpose of planning, each batch p_i , from the set P is defined by its size $size(p_i)$ and *type* (liquid, bulk, etc...) that defines the resources that may carry it. By definition, all batches can be split and re-assembled during transport. In such case, we denote $p_i^{a_j}$ the part of the batch allocated to action a_j .

The transport problem is being solved by **agents** from the set $Ag = \{A_0, A_1, \dots, A_k\}$. As already suggested, the problem is defined by agent A_0 who seeks help from other agents in the community – this agent is a *coalition leader*. Each agent A_i is characterized by its trustfulness $\Theta_{A_j}(A_i)$ as it is perceived by agent A_j , and its *distrustfulness* $\Delta_{A_j}(A_i) = 1 - \Theta_{A_j}(A_i)$. (see Section 5.4 for details)

Agent is therefore modelled by coalition leader the as a tuple $A_i = (\Theta_{A_0}(A_i), res_{A_0}(A_i))$, where the set $res_{A_0}(A_i)$ models leader's knowledge about agent's resources.

⁴ Therefore, in our graph, the nodes are defined as $Ac \cup O$, while the directed edges describe the relations expressed in *allows* and *prer* sets of each action or objective. We may also note that the global state of the system is defined by the state of all objectives.

Besides trust, each agent controls one or more **resources** as defined in its set res_{A_i} . All resources, regardless of the agent they belong to belong form a set $R_{A_0} = \{r_1^{A_i}, r_2^{A_j}, r_l^{A_j}\}$, where the super index of each resource denotes the agent to which this specific resource belongs. Each resource is described by a tuple $r_i^{A_j} = (A_j, allowed_{r_i}, pr_{r_i}, cap_{r_i})$, where the A_j denotes the owner agent of the resource, $allowed_{r_i}$ is a set of actions (transports) to which the resource can be assigned, pr_{r_i} is a set of batch types it can carry and cap_{r_i} denotes a capacity of the resource.

Tasks are a result of the planning process. They form a set $T = t_{a_1}, t_{a_2}, \dots, t_{a_m}$, and each task corresponds to one action. Task is defined as $t_{a_i} = (batch_{t_{a_i}}, com_{t_{a_i}})$, where $batch_{t_{a_i}}$ is a set of batches transported in the task and $com_{t_{a_i}}$ is a set of **commitments** – each commitment⁵ $c = (a_i, A_j, r_k^{A_j}, p_l^{a_i}, cap)$ is an assignment of a specific resource r_k (and consecutively its owner A_j) to one partial batch $p_l^{a_i}$ from the set $batch_{t_{a_i}}$ and cap determines the capacity that is to be assigned. If the r_k capacity allows it, one resource can be committed to more than one batch/action and a single partial batch $p_l^{a_i}$ can be covered by several commitments – in such case, we denote $cap(r_k^{a_i})$ the aggregate size of all commitments from the task t_{a_i} to which the resource r_k is committed. Commitments of resources relative to a single task define a **team** from the set $E = e_{a_1}, e_{a_2}, \dots, e_{a_m}$. The team is simply a subset of the set Ag containing all the agents contributing their resources to the task t_{a_i} . **Coalition** Co is then simply defined as a union of all teams from the set E .

Quantity	Definition	Set
Objective	$o_i = (prer_{o_i}, allows_{o_i})$	O
Action	$a_i = (prer_{a_i}, allows_{a_i})$	Ac
Batch	$p_i = size(p_i)$	P
Agent	$A_i = (\Theta_{A_0}(A_i), res_{A_0}(A_i))$	Ag
Trustfulness	$\Theta_{A_0}(A_i)$	Θ_{A_0}
Distrustfulness	$\Delta_{A_j}(A_i) = 1 - \Theta_{A_j}(A_i)$	Δ_{A_0}
Resource	$r_i^{A_j} = (A_j, allowed_{r_i}, pr_{r_i}, cap_{r_i})$	R
Task	$t_{a_i} = (oper_{t_{a_i}}, com_{t_{a_i}})$	T
Commitment	$c = (a_i, A_j, r_k^{A_j}, p_l^{a_i}, cap)$	C
Team	e_{a_1}	E
Coalition	$coalmem$	Co

Table 7.1. Problem notation summary.

7.4.1 Public, Semi-Private and Private Information

Sharing the information about resources, plans, goals and intentions is significantly different from the cooperative agent systems. In adversarial environments, agents must seriously consider the possibility of information misuse and try to find the equilibrium between minimum information disclosure and cooperation and planning efficiency. Therefore, following [51] and respecting the definitions provided above, we recall three types of information:

Public information is accessible to any agent in the system. It includes information about agent identity, existence, location and basic annotation of provided services - *type* of the resources $res_{A_0}(A_i)$ it offers, but without any information concerning their capacity, number or restrictions. Semi-private information facilitates the planning process. It is mutually shared within groups of trusted cooperators that collaborate frequently and enables them to prepare the plans easier than by negotiating through all possible options [53]. For each agent A_i , we assume that it

⁵ Formally, until being evaluated and updated by bidding agents, commitments must be regarded to as mere *commitment opportunities*.

includes the information about its resources (capacity) aggregated by type and including the restrictions regarding their use on the set Ac .

Private information is reserved only to the owner agent and never shared with anyone else - it contains the detailed information about its resources, including their individual capacity, restrictions, locations and other information.

Note that the sets R as perceived by various agents are not identical due to the fact that they don't have the access to the same information.

Typically, agents will disclose aggregate or approximate information about their resources as a semi-private information. Such compromise provides enough knowledge for the first stage of the planning process, and detailed task allocation is then finalized in course of negotiation without exposing more data than necessary.

7.5 Algorithm Description

This section provides an overview of the planning algorithm we suggest, combining the social model and linear programming planner with focused and well-targeted negotiations in the later stages of the process. The complete planning process is below and detailed in the remainder of this section.

1. **Initial Planning:** Team leader uses its social knowledge and planning capabilities in order to prepare initial plan. This happens in two phases:
 - abstract plan construction and
 - task allocation to the agents.
2. **Local Plan Evaluation:** Initial plan is evaluated by the respective agents:
 - initial plan is received by the perspective members,
 - the members evaluate the plan and make an attempt to trade the commitments within teams,
 - proposals are sent by members back to the leader.
3. **Coherence & Verification:** Proposals are included as an input for the detailed planning, that ensures the plan coherence.
4. **Plan Execution:** Revised commitments are received by coalition members, may be swapped and the plan is being executed.

Plan phases are also detailed in Fig 7.4.

7.5.1 Initial Planning

In the first phase of the plan, we assume that the coalition leader A_0 has obtained a specification of the goals to accomplish and is obliged to form a coalition with other agents in its social neighborhood to accomplish it⁶. It uses its social knowledge about these agents to draft a preliminary plan in the following steps.

Constructing the Abstract Plan. The first step is a preparation of the abstract plan – an action-objectives bipartite graph capturing the relationship between initial and terminal objectives (states) – typically covering alternative solutions. The graph must contain at least one path connecting the initial and terminal objective – if such path can't be identified, agent A_0 is unable to solve the planning problem.

Constructing the abstract plan is a computationally exponential problem in complex domains. Recent advancements in the field of AI planning provided very efficient techniques for constructing the plans in reasonable amount of time such as GraphPlan [45] or SAT-Plan [7]. These techniques implement a sophisticated breadth-first search based on expansion of the bipartite graph or iterative

⁶ In this report, we only consider a single problem planning, while the other problems interact with the current one through the use of resources from the set R and by using the same trustfulness values Θ_{A_0} .

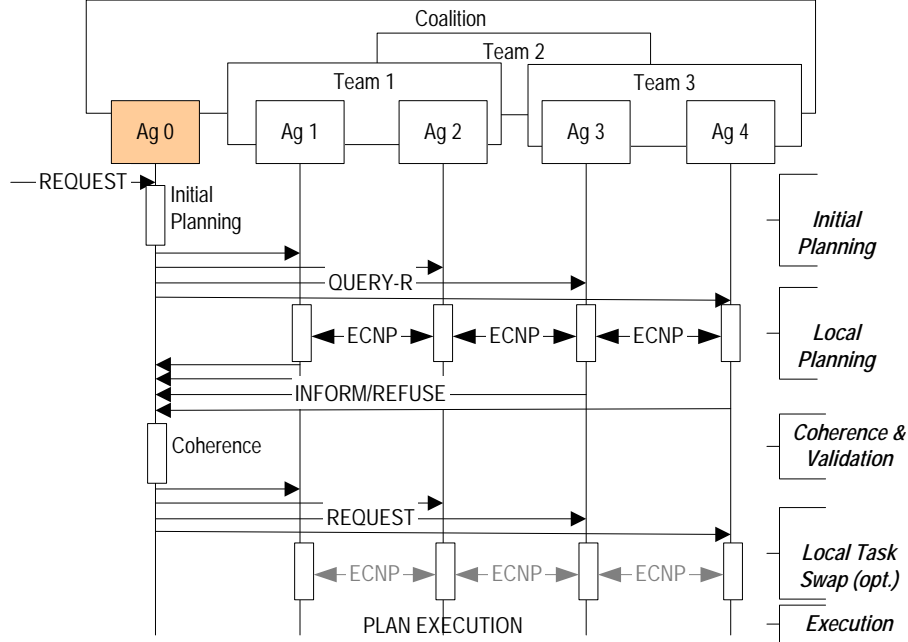


Fig. 7.4. Overview of the protocol phases: Agent A_0 is a coalition leader and has decomposed the global task into three tasks and three teams.

propositionalization of the planning problem. There is an interesting property of GraphPlan that the algorithm can find in a polynomial time whether there exist a solution to the planning problem.

Task Allocation. Once an acceptable abstract plan is established, leader proceeds with the (ii) allocation of batches and resources to individual actions in the plan, while respecting the constraints defined in the objectives. Note that for sake of computational efficiency, some actions and objectives from the abstract plan can be removed during this phase if there are no resources or batches to allocate to them. To allocate the others, coalition leader solves a fuzzy linear programming problem that performs the allocation using the data from the social knowledge and goal specification.

Planning Problem Definition. Use of the fuzzy linear programming (FLP) either provides an acceptable rough task allocation T , or identifies a constraint that prevents the agent from finding the solution, a feature that is crucial in changing environment.

The constraints we define for the problem are the following. The first equation expresses the node equilibria - conservation of goods in each node.

$$\forall o_i \in O \setminus \{o_0, o_n\}, \forall p_j \in P : \quad (7.1)$$

$$\sum_{a_k \in \text{prer}(o_i)} \text{size}(p_j^{a_k}) \cdot \Theta_{a_k} = \sum_{a_l \in \text{allows}(o_i)} \text{size}(p_j^{a_l})$$

where the Θ_{a_k} represents the estimated action trustfulness (see Section 5.4) taken from the trust model (e.g. delivery ratio in our case) - it allows us to model the probable losses in the actions from the set $\text{prer}(o_i)$. Depending on the context, the coefficient may range from 0 (no hope of delivery and therefore no need for subsequent transport) to 1, resulting in the same amount of resources allocated for outgoing cargo.

The initial node has a simpler relation, declaring that we can't take away more cargo than available:

$$\forall p_j \in P : size(p_j) \geq \sum_{a_l \in allows(o_0)} size(p_j^{a_l}) \quad (7.2)$$

while the terminal node doesn't introduce any constraint.

Furthermore, for each action a_i (elementary transport) and each batch p_j , we must ensure that the commitments cover the whole partial batch $p_j^{a_i}$ ($size(p_j^{a_i})/leqp_j$) due to the possible parallelism):

$$\forall a_i \in Ac, \forall p_j \in P : p_l^{a_i} = \sum_{c \in com_{t_{a_i}} : batch(c)=p_l^{a_i}} cap(c) \quad (7.3)$$

then, we must also make sure that no resource is used beyond its capacity:

$$\forall r_i \in R : cap(r_i) \geq \sum_{a_j \in Ac} cap(r_i^{a_j}) \quad (7.4)$$

besides these restrictions, we need to set-up the **utility function** for which we optimize:

Utility functions are highly domain dependent and we present the one we use as an example – we **maximize** the amount of the cargo delivered to the target given the fixed initial batch sizes;

$$U_m = \alpha \cdot \sum_{p_i \in P} size(p_i^{o_n}) - \beta \cdot \sum_{c_j \in C} size(c_j) \Delta_{A_j}(ag(c_j)) \quad (7.5)$$

, where $p_i^{o_n}$ denotes the part of the batch p_i delivered to the terminal objective and $ag(c)$ the agent committing to c .

We minimize the expected amount of the cargo lost (second sum) and we balance the cost of losses and value of delivery (first sum) by setting the constants α and β to domain-appropriate values. The ultimate goal is to allocate the resources of the coalition members to cover most of the delivery, while minimizing the risk of the attack. In an alternative, goal-driven utility function, we may **minimize** the size of the initial batch while covering the predefined request from the terminal locations. This implies also a formal redefinition of the relation 7.2 to cover the terminal node instead.

Transformation of the fuzzy linear problem into the classical one is discussed in Section 7.6.2.

Once the solution of the planning problem is identified, leader determines all perspective coalition members (owners of resources assigned to various tasks) and queries each perspective member whether it is capable and willing to solve the assigned tasks in a given coalition/team. Therefore, each perspective coalition member A_i is sent a following structure: $cma_{A_i} = (A_0, coalmem, assign)$, where A_0 is a coalition leader, set $coalmem$ lists all coalition members and set $assign$ lists the relevant information about tasks the agent's resources are assigned to, defined as $(e_{a_j}, com_{t_{a_j}}(A_i))$, where j is an action (task) index and $com_{t_{a_j}}(A_i)$ are commitments suggested to agent A_i on task t_{a_j} .

7.5.2 Local Plan Evaluation

When the coalition members A_i (selected by the leader in the previous step) receive the coalition proposals from the leader, they must use their private knowledge to create the bid reflecting their preferences and local situation. Several classes of problems must be addressed: **(i)** agents must decide whether they trust the coalition leader and members sufficiently to cooperate with them, typically putting emphasis on their trust in the leader ($\Theta_{A_i}(A_0)$) and agents within the same teams ($\forall e_k : A_i \in e_k \forall A_j \in e_k \Theta_{A_i}(A_j)$). If the agent is confident enough with the coalition and proposed commitments, it will try to assign its resources to its commitments.

At this level, we handle several issues that are ignored by the leader's first-level planning – resource granularity (unknown to the planning agent due to the privacy issues) and relations between the resources assigned to different tasks. In the first round, each agent assigns its resources to the commitments that are the best fit for available resources, trying to cover all commitments. Then, it will offer the excess capacity of the resources assigned to the task t_{a_j} to all members of the team e_{a_j} using the multi-phase auction mechanism described in [28]. This step is designed to eliminate the resource allocation inefficiencies that are due to the possible leader's lack of knowledge about actual resources or a side effect of selected planning method. More formally (see also Fig. 7.5), to start the negotiations, each agent A_i working on task t_{a_j} broadcasts a CFP message containing its free capacity to all team $e_{t_{a_j}}$. If the other team members are interested in using this capacity for the task they were allocated, they submit their bid. Agent A_i selects one or more bids and answers them with a temporary grant, making them binding *for the bidders*; other are temporarily refused. When the agent A_i participates in several teams, it can now reshuffle its resources between the tasks to use them in an optimal manner. Once the resource reallocation is terminated, all compatible temporary grants are confirmed, while the others may be refused (In case we the agent has replaced the original resource with a lower-capacity one.). If appropriate, agent can now offer the new free capacity for trading using the same protocol.

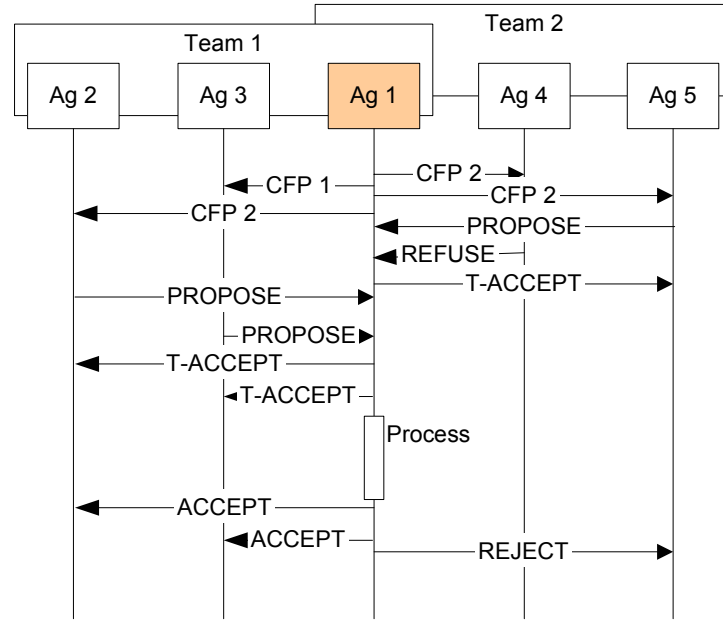


Fig. 7.5. Use of the ECNP to allocate agent's resources across two different teams. Agent A_1 first temporarily accepts the offer from A_5 , but later on finds a better resource allocation and prefers to commit larger resource to team 1. Therefore, it rejects the bid from A_5 .

Note that the auctioning and negotiation takes place only within the single task team, therefore minimizing the knowledge dispersion and communication load. On the other hand, agents may therefore miss a better task allocation. Once the negotiation is finished, all team members prepare and submit their answers to the coalition leader. The answer is composed of by the list of commitments that are actually *binding* for each agent, but may differ from those originally assigned to the agent as: (i) the agent is not always able to cover the whole assigned commitment and commits only to

a part of the original commitment or (ii) it notifies the coalition leader about the transfer of the whole commitment or its part to other coalition member (this member lists this commitment in its turn as covered).

When the agents submit their binding commitments to the coalition leader, they have an alternative to offer the free capacity of the resources they've allocated to the task to the coalition leader - the leader may include use it to cover other batches from the same task, as specified by relation 7.6. While this remains an attractive optimization feature, this approach has two major drawbacks - the leader can easily guess the capacity of agent's resources and the free resources can not be used on another task. Therefore, we consider the member's reasoning and negotiation within teams as crucial part of the algorithm, effectively merging the decision-theoretic approach from the first and third step with traditional agent approaches.

7.5.3 Coherence & Verification Phase

In this phase, coalition leader receives the answers from the coalition members and must re-combine them into a globally coherent plan. As the initial planning has produced a coherent plan, the plan is coherent when all proposed commitments were covered by members. If not, the leader must add all updated commitments/refusals from the agents to the initial plan and perform the new calculation to make sure that the condition 7.1 is valid for the final plan.

Updated commitments are included as follows (refusals or previously unassigned commitments are considered as commitments with 0 capacity):

$$\forall a_i \in Ac, \forall r_j \in R : cap(r_j^{a_i})^{prop} \geq cap(r_j^{a_i})^{final} \quad (7.6)$$

In this phase, the FLP algorithm may be unable to find a solution. In this case, besides abandoning the project altogether or trying to recruit more agent, coalition leader may simply re-state the project with larger teams assigned to the tasks; even if some team members have no initial proposed commitments, their inclusion opens more negotiation options. On the other hand, when many agents refuse to commit any resources to some task, coalition leader may deduce that some other team member is probably considered as dangerous and be more restrictive in collaborator selection.

If the coalition leader manages to find an acceptable planning outcome, it prepares the final commitments (with the quantities assigned that are less or equal to the binding ones proposed by members) and re-submits them to the coalition members.

7.5.4 Plan Execution

As the proposals by the agents were binding, coalition members shall be all able to start performing the assigned tasks immediately. Alternatively, when the final commitments are lower than the ones they have proposed, they may change their resource allocation or trade the assignments with their peers in the team in the same way as in the Local Plan Evaluation phase, provided that they manage to honor their commitments.

7.6 Algorithm properties

In this section, we will analyze the above-described algorithm and discuss several interesting properties it presents: computational efficiency, preservation of private information and stability of the solution with respect to environmental perturbations.

7.6.1 Overall Characteristics

The claim that the algorithm we propose differs from previous approaches due to the following properties:

- **Reduced communication** is a result of the use of the social knowledge in the *initial planning* step of the algorithm. Instead of several rounds of auctions, action decomposition and backtracking, coalition leader uses its social knowledge to compose balanced task teams and pre-assign pre-commitments to each potential coalition member.
- **Operation-research integration** with negotiation and cognitive methods is natural and seamless: output of most trust models in existence today can be transformed into the fuzzy number-form and this representation fits well into the context of modern FLP methods as shown in Section 7.6.2. On the other hand, the individual team members don't need any notion of FLP techniques - they only reason about the coalition, their teams and negotiate within their teams to achieve optimal resource allocation.
- **Contraction** of the solution space is another key feature – each step of the planning, centralized or distributed, reduces the solution space. Initial planning performs the greatest reduction, as the actions/tasks are selected, resources pre-allocated and agent teams created. Local planning phase then further clarifies resource allocation and team composition and the results of this phase are incorporated as additional restrictions for the FLP planning problem solved in the coherence and validation phase – we effectively ensure that any overall solution will respect the commitments received from coalition members and can be executed. Optional re-allocation step doesn't break our assumptions, it only permits team members to trade resources in a situation where the batch sizes may have been reduced. If the plan can not be implemented due to the member refusal or resource incompatibility, the situation is detected in the coherence planning step. LP method used identifies the interfering restriction and can direct the coalition leader towards plan reconfiguration. Therefore, in the global algorithm as suggested, we don't allow any backtracking (except the team-scale negotiation), increasing the outcome predictability. On the other hand, the algorithm as presented doesn't guarantee that the result it returns will be the optimal plan. We don't consider this as a serious drawback, because none of the comparably efficient algorithms currently in use can guarantee such result.

7.6.2 Stability of Flexible & Fuzzy Linear Programming

One of the important properties of the trustfulness $\Theta_{A_0}(A_i)$ (and distrustfulness $\Delta_{A_0}(A_i)$) values is their uncertainty, emphasized by the fact that they are modelled as fuzzy numbers. There are two approaches how to use these values in the LP algorithms: either to solve a *flexible linear programming* problem, or to *defuzzify* the values and solve a classical LP problem.

Flexible linear programming techniques [15] that work with fuzzy coefficients provide us with a unique feature - a stability of the solution with respect to small changes of coefficient (e.g. trustfulness) values defined as symmetrical triangular fuzzy numbers. Problem formulation remains the same, but we must solve a non-linear optimization problem in order to obtain the solution – a major disadvantage of the approach. On the other hand, once we have an appropriate solver, we may effectively adjust the stability of the solution by varying the width of the trustfulness values – by restricting their width, we approach the unstable classical linear programming problem, while the widening of trustfulness representation ensures the stability with respect to bigger perturbations. This ability is a very desirable feature when the agents encounter an intelligent adversary in an unknown environment – agents can adjust their planning to be robust when they still gather the information about the environment and reduce the predictability of their behavior in later phases. The shape representing the trustfulness $\Theta_{A_0}(A_i)$ supports this adaptation, as it "narrows" with the increasing number of data.

In the alternative approach, $\Theta_{A_0}(A_i)$ and $\Delta_{A_0}(A_i)$ must be defuzzified before they are inserted into the planning constraints of the normal LP problem. Defuzzification to use depends on the definition of \leq relation between fuzzy numbers – as we follow the FLP \rightarrow LP transformation method detailed in [15] we use the center of gravity to compare two fuzzy numbers. This relation between fuzzy numbers returns a *crisp* output. During the transformation, we replace the fuzzy numbers in the constraints and utility function by the center of the core of the trustfulness values and solve the resulting problem.

We may also defuzzify the $\Theta_{A_0}(A_i)$ and $\Delta_{A_0}(A_i)$ using the center of the core method - this approach is more sensitive to the noise, especially with limited number of data, but as the agent gathers more experience, it converges to the center of gravity method due to the shape of the trustfulness function $\Theta_{A_0}(A_i)$ as defined in Section 5.4.2.

7.7 Conclusions and Future Work

In this section, we have presented a combined planning algorithm that can be used to efficiently create a shared plan in an adversarial environment, featuring only a limited and controlled information disclosure by self-interested agents. Adversarial behavior of agents and environmental reasons of failure (actions with low action trustfulness) can be detected and provide an input for the embedded trust model, that in its turn provides an input for further planning.

One of the important open issues of this research topic is the concept of *plan diagnosis*. Plan diagnosis [77] is important to achieve long-term efficiency by elimination of untrustful cooperators and bad actions (paths). We can not assume that the state of the problem is observable – implicitly, we assume that only the initial and terminal objective status are known by the coalition leader and that the state of some of the intermediary objectives **may** be known. Integration of the latest results from the monitoring selectivity problem [39] into a trust model update is a challenging problem for future research.

Another key challenge for the future research in this area is to investigate the plan execution phase and allow *intelligent replanning*. Re-planning can occur as a result of a coalition leader detects a failure in completion of one or more commitments or upon a request from the coalition members. If these commitments fully or partially pre-condition other tasks (and their commitments), it may be advantageous to re-plan the plan in order to eliminate inefficient future commitments. Such operation is analogous to coherence phase (see Section 7.5.3), but for each commitment with known outcome, we can fix the commitment size to the real delivered value, or limit it by using the information about partial deliveries/losses. In the same manner, the agent can react when some task was more successful than expected and more resources are necessary for subsequent transport. Integration of the classical work on joint commitments [18] and shared plans [33] as well as various penalty mechanisms allowing agents to deliberate on dropping the commitments is an important component of the future research in this area.

8

Conclusions and Future Work

In this document, we have presented the results of the research project: Advanced Agent Methods in Adversarial Environment funded by AFRL/EOARD under the contract number FA8655-04-1-3044. The project has both engineering and research results, as presented in respective chapters. This report has been structured to reflect the results from all research phases, starting with fundamental issues – *formal model of adversariality*, until now missing in the agent community and *formal model of inaccessibility* that were defined and published.

These results enabled us to work on related practical research problems: the definition of the *trust model* that is suited for autonomous lightweight devices in potentially adversarial environment – our model provides unique features like noise robustness, automatic environmental adaptation, uncertainty-inclusive reasoning and being iterative, it is very computationally and data efficient. On the other hand, it is sufficiently rich to be used as an input for planning mechanisms.

We have also analyzed and proposed several inaccessibility solutions, most notably the *stand-in agents* that are well suited for non-collaborative environment where the potential middle agents are not necessarily trusted. Furthermore, as we have noted that the efficiency of inaccessibility solutions is a key enabler for their use, we have defined a universal middle agent architecture integrated with social-dominance-based optimization model. This model enables *fast adaptation* of middle agent network (regardless of the agent type) using only *local knowledge* – no central coordination mechanism is necessary and the network still adapts to the changes in the environment to reach the optimum number of agents relatively fast. Message/knowledge routing is also optimized in a similar manner, using the virtual micro-payments that guide the system self-optimization.

Both the trust model and inaccessibility solutions were rigorously verified in the experiments. The results of these experiments provided us with the knowledge necessary for the integration of both models into a coherent system, solving the *distributed logistics problem* in an adversarial and inaccessible environment. This problem is implemented by a specific humanitarian relief scenario realized in ACROSS framework in **A-globe** platform. The core feature of this system is the *planning algorithm* that integrates negotiation and fuzzy linear programming based planning into efficient system that is well suited for adversarial environments – trust both in the agent and actions is a crucial planner input, as well as the social knowledge. Both the classic social knowledge and trust model data is updated via the adaptive stand-in network, and the stand-ins are also responsible for the goal identification.

All the development and the experiments were realized in **A-globe** multi-agent platform using the ACROSS scenario. During the project, the **A-globe** platform ¹ was significantly enhanced to handle bigger number of agents and increased message flow in the simulated community. The platform, in a conjunction with the scenario were successfully presented and were awarded two awards at

¹ Whose development was originally funded by AFRL/EOARD project number FA8655-04-1-3044.

international conferences: CIA 2004 System Innovation Award and The 2005 IEEE/WIC/ACM WI-IAT Joint Conference Best Demo Award.

This project has delivered well characterized and tested mechanisms that are available for reuse: the trust mechanism, generic stand-in self-optimizing architecture and planning mechanism for adversarial environment. Some of the considered options for future extensions are:

- **Teamwork** in adversarial environment based on an efficient planning can be used not only in logistics domain, but can be also used in a more general manner: flexible team formation between UAVs that provide protection, communication and observation to ground vehicles is one of the considered application domains. Such technique would allow autonomous tasking and flight of the UAVs (and also planning for ground vehicles), reducing the load on the operator and allowing it to handle exceptional situation only.
- **Trust model** is being developed to integrate better reputation mechanism, information decay in time, context-dependent trust and other important concepts. All these techniques will be combined with the core model only if appropriate and necessary, as they increase the computational complexity. Currently, we are investigating the use of the model in the embedded security system for network infrastructure and the extensions necessary for this domain.
- **Stand-in** infrastructure will be further developed as a secure method how to ensure synchronization in an inaccessible environment. Emphasis will be put on the optimization mechanism fine-tuning. We want it to automatically adapt not only to changes in the environment, but also to perform meta-adaptation: adaptive mechanism will change its parameters to match the dynamics of the environment to avoid local overfitting or insufficient adaptation.
- **Social dominance models for optimization** that are currently used for stand-in network optimization can be used in completely different domain altogether. One of the possible applications is a development of a robust reputation mechanism to counter the strategic behavior in collective trust modelling. Another application is a self-organization of flight formations between autonomous aerial vehicles to facilitate airspace management and deconfliction.
- Research in **formal models of adversariality** and their practical applications in negotiation and adversariality detection.
- Research in **formal models of accessibility** and network characteristics, using the results from the random graph theory. These results will be useful for the evolution of stand-in optimization mechanism.

A

Progress in **A-globe** Multi-Agent Platform Development

In this section will be in brief described **A-globe**, an agent platform designed for fast prototyping and application development of multi-agent systems. **A-globe** provides the same level of services as JADE, COUGAAR, FIPA-OS, JACK and in addition has some special features focused on simulation of the real-world dynamic environment. The main focus of the **A-globe** developers has been given to the following applications domains:

- **simulation**, especially simulation of the multi-agent environment and collective behavior of large communities
- **scalability**, high-number of fully autonomous agents, that are loosely coupled with *lightweight infrastructure*
- **agent migration**, persistence and code and state migration within the communication network as much as physical reallocation of the computational host and thus modelling of partial and non-permanent *communication inaccessibility*[49].

The platform provides functions for residing agents, such as communication infrastructure, store, directory services, migration function, deploy service, etc. Communication in **A-globe** is very fast and the platform is relatively lightweight.

A-globe is suitable for real-world simulations including both static and mobile units (e.g. logistics, ad-hoc networking simulation), where the core platform is extended by a set of services provided by Geographical Information System (GIS) and Environment Simulator (ES) agent. The ES agent simulates dynamics (physical location, movement in time and others parameters) of each unit.

A.1 System Architecture

The system integrates one or more agent platforms. The **A-globe** design is shown in Figure A.1. Its operation is based on several components:

- **agent platform** – provides basic components for running one or more agent containers, i.e. container manager and library manager;
- **agent container** – skeleton entity of **A-globe**, ensures basic functions, communication infrastructure and storage for agents;
- **services** – provide some common functions for all agents in one container;
- **environment simulator (ES) agents** – simulates the real-world environment and controls visibility among other agent containers (section A.2);
- **agents** – represent basic functional entities in a specific simulation scenario.

Simulation scenario is defined by a set of actors represented by agents residing in the agent containers. All agent containers are connected together to one system by the GIS services. Beside

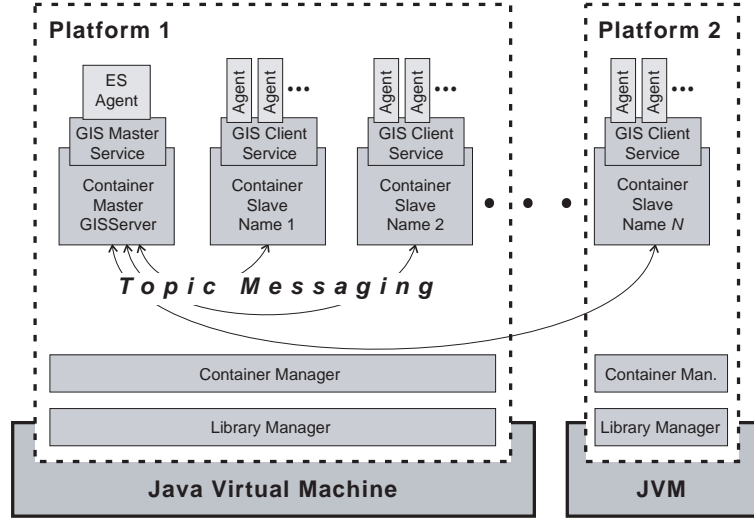


Fig. A.1. System Architecture Structure

the simulation of dynamics the ES agent can also control communication accessibility among all agent containers. The GIS service applies accessibility restrictions in the message transport layer of the agent container.

A.2 Simulation Support in \mathcal{A} -globe

While designing the simulations in \mathcal{A} -globe platform, we use agents not only to play roles in the simulated world - *actor agents* but we also use them to implement the world where the actor agents act. The agents used for the world simulation are all located in a dedicated master container and are called *Environment Simulation agents*.

These agents only rarely use messages to communicate with actor agents. Instead, they communicate with *topic messaging* - container-to-container messaging specifically reserved for environmental simulation.

Topic messaging is managed by GIS Service - a special service that is a part of the \mathcal{A} -globe platform and can be started in a container by specifying an appropriate startup parameter. This parameter value determines whether the container is a *master*, server side container or a *client* - normal container with actor agents.

Client agents subscribe with GIS client service to receive various topics. If such a topic is received by the container, it is distributed to all subscribed agents. Note that all agents in the container receive the same value - this is appropriate in our opinion, as the environment perception shall be identical for all collocated agents. In addition, the agents who wish to act on the environment can submit topics to the GIS service. These topics are then sent to all ES agents in the master container subscribed to receive the topic.

In the nominal configuration, each ES agent manages an internal model of the environment, updates the model with the actions received from actors and submits the environment status to actors in their containers. Each ES agent can handle one or more topics and one topic can be handled by more than one agent. Specialized ES agents can also subscribe to receive local topics from other ES agents. Typically, many specialized ES agents can receive position information from position agent and use this data to submit appropriate localized environment information to agents.

This approach scales fairly well with the community size. However, when the environment becomes more complex, it is often not economic to handle the environment simulation in the ES agents

as the interactions become too cumbersome and internal models too complicated. In this case, the server can use an appropriate GIS server with an ES agent wrapper for simulation purposes. The ES agent(s) is then responsible only for obtaining the information from the appropriate layers of the GIS server and submitting them to corresponding topics. The use of the GIS server is not without a cost - the integration with wrapper agent is rarely flawless and shall be avoided for simple environments.

ES agent can be responsible for nearly any simulation layer, depending on the wishes of the developers. However, a privileged place between ES agents is occupied by *accessibility agents*, who control the existence of communication links between containers holding the actors. Their prominence is caused by the fact that the platform messaging layer is integrated with these agents through predefined topics and any attempt to send a message to an agent in an inaccessible container is automatically unsuccessful.

There are several ES agents implemented and some of them are provided as a part of the **A-globe** platform package for optional use:

- **Manual (Matrix) ES Agent** This agent provides simple user-checkable visibility matrix. The user simply checks which containers can communicate together and which can not.
- **Distance-based ES Agent** This agent is a fully automatic environment simulator. It receives positions of mobile agent containers representing mobile units in virtual world and automatically controls accessibility between them. The visibility is controlled by means of the simulation of the short range wireless link. Therefore each container can communicate only with containers located inside the predefined radius limit. As the containers move, connections are dynamically established and lost.

Other visibility agents can be implemented for each specific simulation, provided that they respect the ontologies and protocols that apply for them.

A.3 New features and Enhancements in **A-globe** version 2.1

The agent platform **A-globe** ver 2.1 is a major feature release. The features listed in this section are introduced in ver 2.1.

A.3.1 **A-globe** upgraded to the JAVA 2 edition 5.0

The JAVA 2 edition 5.0 [69] or later is required for running **A-globe** ver 2.1. New components of **A-globe** agent platform use new JAVA language features, such as *generics*, *enhanced for loop*, etc. **A-globe** platform also uses runtime JVM improvements, such as *garbage collection ergonomics*, *string builder*, *opengl*, etc.

A.3.2 Library version handling

The Library Manager component is no longer integrated with the platform, but is now an integral part of agent container.

Every new loaded library in **A-globe** container internally uses the library name constructed from the original library name and SHA-1 hash [1] of the library content. The loaded library is automatically labelled with unique version label constructed as `ver{ver_num_in_the_container}@{container_name}`. In such a way, two different libraries with same file name can be used in parallel within single **A-globe** platform. The library can be removed before a class loader opens it. After opening, it can not be removed from the runtime environment. It can be removed at **A-globe** restart.

A class loader is defined for each agent and service. If an agent/service doesn't use any special library, it uses a bootstrap class loader. The bootstrap class loader locates classes only in the name

space defined in the path specified by the starting **CLASSPATH** parameter or manifest **CLASSPATH** parameter in the JAR library used by java runtime. If an agent/service uses one or more specific libraries, it has to define its own class loader which tries to load classes in specified libraries. An agent specific class loader always prefers classes defined by the bootstrap class loader. Therefore, default classes can not be "overridden" - it has no sense to define own `java.lang.String` class because it will never be actually loaded. In **A-globe**, each agent(service) class loader defines agent/service class resolving *name space*. The migration process and the message transport layer always use respective *name space*.

In one agent container, several agents with same main class but different class versions can run. For example, there can be several migrating agents using `examples.agent.migrating.MigratingAgent` main class, but each agent can use its own library with different class definition.

A.3.3 Message Transport layer

The previous release version of **A-globe** featured *message transport* component in agent container only. All messages sent between two agent containers running in a single agent platform were sent via TCP/IP stream. Now, the new platform-level *message transport* component ensures more efficient exchange of these messages. This enhancement improves messaging speed between agent containers within one Java Virtual Machine (single platform).

The *message transport* layer takes care that all message are serialized (marshaled) and deserialized (unmarshaled) in the appropriate class name space depending on the sender and receiver agent/service's class loader.

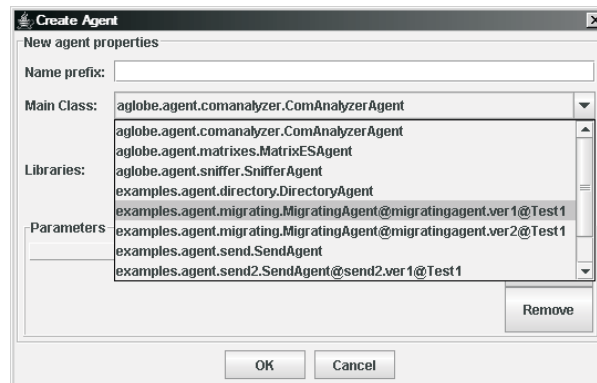


Fig. A.2. Create new agent dialog with Class Finder results

A.3.4 Topic messaging

Topic messaging is managed by GIS Services - a special services that are a part of the **A-globe** platform. Topic messaging were described in section A.2.

A.3.5 Class Finder

The *Class Finder* component of **A-globe** locates all available classes that can be loaded as agent/service when new agent or service dialog is shown. It provides list of main classes for easy startup of new agent/service. When an agent/service is defined in specific loaded library and that library is necessary for its execution, the required library is specified behind the character '@' as shown on figure A.2.

A.3.6 Directory Service

The *Directory Service* provides extended white pages and directory pages services. It supports inaccessible environment and uses visibility updates provided by ES Visibility servers. The service is automatically started on every client container. Previous version of directory service is no longer used in **A-globe** ver 2.1.

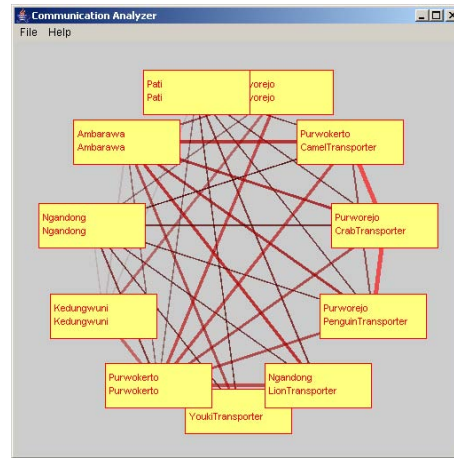


Fig. A.3. The Communication Analyzer

A.3.7 Sniffer Improvements

There are several enhancements in the sniffer agent since previous **A-globe** release:

- **Message history length** – this parameter is useful in large-scale scenarios where thousands of messages are sent in short time period. Unnecessary outdated messages in the history requires more memory for sniffer agent.
- **Agent/service address filter** – every new agent or service actor address known by sniffer agent is matched against that regular expression. If test is positive, the actor is automatically labelled as visible in the message detail window. This feature is useful when someone wants to sniff only message between a limited group of agents and services, e.g. between transporter agents. Default filter is set to the *others*, showing all agents and messages.
- **Browsing through the conversation** – two new buttons are available in the message detail window for moving forward and backward in the conversation history.

A.3.8 Communication Analyzer

The *Communication Analyzer* provides view of the intensity of interactions between agents, see figure A.3. *Communication Analyzer* presents selected agents, filtered by a regular expression, in a circle. Messages exchanged between agents influence the color and width of the links between communicating agents. In order to keep the image updated, old messages fade away progressively and only the recent ones are visible. The *communication analyzer* agent can run only in the server container likewise the sniffer.

A.4 New features and Enhancements in **A-globe** version 2.3

Improvement of A-globe message transport layer – low level message processing and transmission in the **A-globe** message transport layer has been changed so that the **A-globe** messaging in byte mode is faster. The **A-globe** message envelope uses externalization with special **String** encoding instead of standard serialization. Externalization may be up to 40% faster than object serialization [71].

Extension of visualizer communication protocol – the **VisioConnection** module used for communication with external visualizers can receive incoming commands from visualizers and distribute them back to the system via **VisioCallbackListener**. This feature can be used in situation when a detailed information about specific object needs to be updated in the external visualizers only on request. This improvement rapidly decreases bandwidth requirements between **A-globe** and visualizers. We use this feature in the Air Traffic Simulator where the detailed information about future flight plans are shown only for limited group of planes, and the updates of detailed flight plans are distributed only for selected planes.

New DataAnalyzer tool – the *DataAnalyzer* tool is useful tool for tracking numeric values that evolve in time, for example while debugging of scenarios or preparing of measurements. Any agent can send its own data for analysis. Data is transmitted to the analyzer by submitting topic **TOPIC_TEST_DATA** to the server with content **AglobeParams**. Example of sending data can be found in the method `aglobe.agent.dataanalyzer.RandomDataAgent.RandomTimer.run()`. Agent is able to show various data from more then one agent simultaneously and filters are available to select only the data that are of interest to observer. This agent runs on master container.

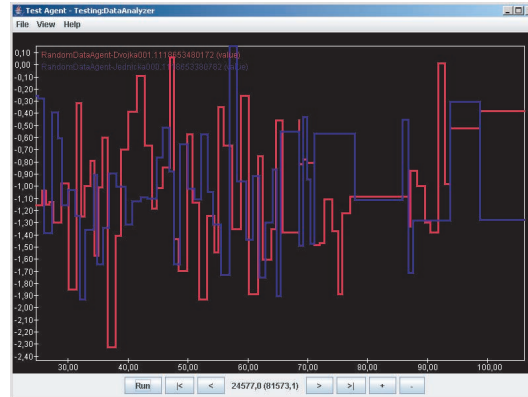


Fig. A.4. **A-globe** DataAnalyzer tool

A-globe ontologies changes – ontologies used internally by **A-globe** were moved from package `atg.ontology` to `aglobe.ontology`. The ontology **Param** and **Params** were renamed to **AglobeParam** and **AglobeParams** to remove many ambiguous conflicts between **A-globe** internal and user-defined ontologies. The library `ontology.jar` is no longer necessary for running **A-globe** and there is no circular reference between ontologies and **A-globe** anymore.

Changed service shell concept – now an agent specifies an service shell owner during request for a service addressed to the *ServiceManager*. The agent no longer needs to take care of unsubscription, deregistration or disconnecting of a service before migration or agent termination, as well as reconnection, resubscription and registration immediately after the migration or cloning.

Agent class update and extension – we have added several convenience methods to simplify the usage of agent cloning/migration and logging functions. An agent who wants create its clone on the same or other agent container needs only call `clone` method with specification of the name

of cloned agent and destination container address. Moreover, the migration procedure of the agent was updated to make it faster. There are new logger methods for easy creation of log messages from within the agent. To create a log message, the agent needs to call `logInfo` or `logSevere` method with single `String` argument. The logging method automatically appends creator's agent name with container specification and timestamp of the message.

Removed bug in Directory Service – in the previous **A-globe** release there was a infrequently occurring problem with distribution of updates when there was more then one receiver of a set of service providers matching with matching filter. This problem was corrected in the current version.

A.5 New features and Enhancements in **A-globe** version 2.4

Recently, we have implemented several new features into the platform: **A-globe web interface** – we have created a specialized service that functions as a web server for agents and containers running on the platform. Using this interface, user is able to manage running agents, start new agents, or load new libraries to the running **A-globe** even from remote location. Agents themselves can create their web pages and are able to post the information on this web server, to the location determined by their container and name. They can also receive the user input from their web pages instead of their GUI, using the standard HTML forms. CSS is supported.

Related feature is the **bidirectional flow between the 3D visualizer and agent**: now, agents can request the data from the visualizer, including the virtual camera view from their environment, actually seeing what happens in the simulated environment. This mechanism allows agents to post this information on their web page in (nearly) real-time, providing an access to the 3D information for remote thin clients.

The implemented mechanism can also be used to obtain richer user information about individual agents – now, the trustfulness of other agents is submitted to the 3D visio by evaluating agent and can be visualized by clicking on agent name in the container panel.

Another major improvement is an inclusion of several **FIPA protocols** into the **A-globe** release. These protocols were developed as a part of **ACROSS** scenario and have now become an integral part of the platform.

Besides usual bug corrections and performance enhancements, two other features are worth mentioning: it is now possible to run the system in a **single platform mode** by using the recently added parameter `-noListen`. This will deny the use of network interface and the performance of the system increases dramatically as the overhead is reduced. On the downside, this mode is useful for single-host running scenarios only¹. The other feature is a possibility to prohibit the store modification by the system, actually **fixing the same agent setup** through the runs. This is useful for the simulations where the consistency between successive experiments is a major requirement.

A.6 Future development possibilities of **A-globe**

In this section, we present several options of future **A-globe** development. However, in contrast to other similar sections in this document, we don't necessarily intend to pursue all the possibilities listed herein. The goal of this section is to stimulate the discussion and to obtain feedback of the user community. If some of the points below are of the interest to you, please let us know.

Lightweight version of A-globe – we can design a lightweight **A-globe** version that may run on mobile devices with very restricted resources (limited memory, slow CPU) such as mobile phones or handhelds. This version of **A-globe** should run on Java 2 Platform, Micro Edition [70]. It will retain basic compatibility with main **A-globe** version – an agent running on this lightweight version

¹ 3D Visio connection and web service functionality is not affected by the parameter.

of **A-globe** will be able to communicate with agents running on standard **A-globe**. Agents should be able to migrate between mobile devices and also between mobile devices and standard **A-globe** platform.

Resource balancing – implementation of **A-globe** monitoring and load balancing module. The monitoring module should acquire information about each agent’s resource consumption (memory and processing time of the host used by an agent where **A-globe** is running). It shall act as a process/task manager in the operating system, but on the **A-globe** agent platform level. The balance module will negotiate with other platforms and may decide to migrate an agent consuming huge part of platform resources to the other host with enough resources. Each agent in **A-globe** will be able to specify if it should be automatically moved, moved after it gives a permission or cannot be moved at all.

Standard simulation tools – we plan to prepare a common set of environmental simulation agents (ES) that can be used for rapid creation of a new simulation scenario. The environmental agents in **A-globe** are responsible for simulation of the environment in the scenario world – for example position and movement of the entities, physical interaction between them, imposing visibility restrictions on the entities, weather information depending on container position in the simulated world, etc. Created set of the EA should support both 2D and 3D worlds. Currently, most ES agents are scenario specific.

Limited communication bandwidth – in the current **A-globe** version the environmental agents cannot control communication bandwidth between entities, they can only control communication visibility. When entities are running on the same host, the communication bandwidth limitation is given by the speed of host CPU. When entities are running on different hosts, the limitation is given by the available network bandwidth (decreased by TCP/IP protocol consumption) connecting these hosts. We can implement new bandwidth control module in the current **A-globe** message transport layer that can impose bandwidth restrictions on link between any two containers in the scenario (unprocessed message can be refused due to insufficient bandwidth or wait in the queue). This feature will enable us to better simulate scenarios with embedded or ubiquitous devices.

Client interaction tool – current **A-globe** version provides GIS topic messaging [48] which is very useful for distribution environmental information between server and client agent containers. Using client interaction tool, an agent will be able to interact directly with other agents. For example, an agent should leave some object or information in its current position and other agents approaching that position can see the object, take that object or use the information. This approach could help us to save the server processing power in distributed simulations.

References

1. FIPS PUB 180-1. Federal standard 180-1: Secure hash standard. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>, 1995.
2. R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, 2002.
3. G. L. Alexanderson. *The random walks of George Pólya*. The Mathematical Association of America, 2000.
4. Joan Ametller, Sergi Robles, and J. A. Ortega-Ruiz. An implementation of self-protected mobile agents. In *ECBS*, pages 544–549, 2004.
5. R. Axelrod. The evolution of strategies in the iterated prisoner’s dilemma. In L. Davis, editor, *Genetic algorithms and simulated annealing*, pages 32–41. Research notes in AI, Pitman/Morgan Kaufmann, 1987.
6. Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.
7. M. Baiocchi, S. Marcugini, and A. Milani. C-satplan: a satplan-based tool for planning with constraints. In *Proceedings of AIPS-98 Workshop on Planning as Combinatorial Search*, 1998.
8. A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
9. Gary S. Becker. Crime and punishment: An economic approach. *The Journal of Political Economy*, 76(2):169–217, 1968.
10. Andreas Birk. Boosting cooperation by evolving trust. *Applied Artificial Intelligence*, 14(8):769–784, 2000.
11. B. Bollobás. *Random Graphs*. Cambridge University Press, 2nd edition, 2001.
12. G. Bortolan and R. Degani. A review of some methods for ranking fuzzy subsets. *Fuzzy Sets and Systems*, 15:1–19, 1985.
13. Sviatoslav Brainov. The role and the impact of preferences on multiagent interaction. In *ATAL ’99: 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL)*, pages 349–363. Springer-Verlag, 2000.
14. W. Cao, C.-G. Bian, and G. Hartvigsen. Achieving efficient cooperation in a multi-agent system: The twin-base modeling. In P. Kandzia and M. Klusch, editors, *Cooperative Information Agents*, number 1202 in LNAI, pages 210–221. Springer-Verlag, Heidelberg, 1997.
15. Christer Carlsson and Robert Fullér. *Fuzzy Reasoning in Decision Making and Optimization*. Physica Verlag, Springer, Heidelberg, 2002.
16. C. Castelfranchi and R. Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, page 72. IEEE Computer Society, 1998.
17. Cristiano Castelfranchi, Rino Falcone, and Giovanni Pezzulo. Integrating trustfulness and decision using fuzzy cognitive maps. In *iTrust*, pages 195–210, 2003.
18. P. Cohen and H. Levesque. Intention is a choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
19. Rosaria Conte and Cristiano Castelfranchi. From conventions to prescriptions - towards an integrated view of norms. *Artif. Intell. Law*, 7(4):323–340, 1999.
20. Rajdeep K. Dash, Nicholas R. Jennings, and David C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

21. P. Davidsson, L. Henesey, L. Ramstedt, J. T?nquist, and F. Wernstedt. Agent-based approaches to transport logistics. In *Workshop on agents in traffic and transportation held in conjunction with the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS04)*, New York, 2004.
22. D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Springer-Verlag, New York, 1993.
23. D. Dubois and H. Prade. Fuzzy real algebra:some results. *Fuzzy Sets and Systems*, 2(4):327–348, 1979.
24. Evaggelia Pitoura Elias Leontiadis, Vassilios V. Dimakopoulos. Cache updates in a peer-to-peer network of mobile agents. In *Proceedings of the Fourth IEEE International Conference on Peer-to-Peer Computing*, pages 10–17. IEEE Computer Society, 2004.
25. James D. Fearon. Rationalist explanations for war. *International Organization*, 49(3):379–414, 1995.
26. J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13. ACM Press, New York, 2002.
27. FIPA. Fipa contract net interaction protocol specification. Foundation for Intelligent Physical Agents, <http://www.fipa.org/specs/fipa00029>, 2002.
28. Klaus Fischer, Jörg P. Muller, Markus Pischel, and Darius Schier. A model for cooperative transportation scheduling. In *Proceedings of the First International Conference on Multiagent Systems.*, pages 109–116, Menlo park, California, June 1995. AAAI Press / MIT Press.
29. Philippe Fortemps and Marc Roubens. Ranking and defuzzification methods based on area compensation. *Fuzzy Sets Syst.*, 82(3):319–330, 1996.
30. FS1037C. Federal standard 1037c: Telecommunications: Glossary of telecommunication terms. <http://www.its.bldrdoc.gov/fs-1037/>, 1996.
31. D. Gambetta, editor. *Trust: Making and Breaking Cooperative Relations*. Basil Blackwell, 1990.
32. Nathan Griffiths and Michael Luck. Coalition formation through motivation and trust. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 17–24. ACM Press, 2003.
33. Barbara Grosz and Sarit Kraus. The evolution of SharedPlans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agencies*, pages 227–262. Kluwer Academic Press, 1999.
34. Afsaneh Haddadi. *Communication and cooperation in agent systems: a pragmatic theory*. Springer-Verlag New York, Inc., 1996.
35. Richards J. Heuer. *Psychology Of Intelligence Analysis*. Center for the Study of Intelligence, US National Technical Information Service, 1999.
36. Jon Himoff, Petr Skobelev, and Michael Wooldridge. Magenta technology: multi-agent systems for industrial logistics. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 60–66, New York, NY, USA, 2005. ACM Press.
37. Dong Huynh, Nicholas R. Jennings, and Nigel R. Shadbolt. Developing an integrated trust and reputation model for open multi-agent systems. In *Proc. 7th Int Workshop on Trust in Agent Societies*, pages 65–74, 2004.
38. Trung Dong Huynh, Nicholas R. Jennings, and Nigel R. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *ECAI*, pages 18–22, 2004.
39. G. A. Kaminka and M. Tambe. Robust multi-agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research*, 12:105–147, 2000.
40. Sarit Kraus, Onn Shehory, and Gilad Taase. The advantages of compromising in coalition formation with incomplete information. In *AAMAS*, pages 588–595, 2004.
41. Milan Mares. Fuzzy coalition structures. *Fuzzy Sets Syst.*, 114(1):23–33, 2000.
42. V. Mařík, M. Pěchouček, and O. Štěpánková. Social knowledge in multi-agent systems. In M. Luck, V. Mařík, and O. Štěpánková, editors, *Multi-Agent Systems and Applications*, LNAI. Springer-Verlag, Heidelberg, 2001.
43. S. Marsh. Formalising trust as a computational concept, 1994.
44. W.Q. Meeker and L. A. Escobar. *Statistical Methods for Reliability Data*. John Wiley and Sons, Inc., 19968.
45. I. Miguel, P. Jarvis, and Q. Shen. Flexible graphplan. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence*, pages 506–510, 2000.
46. Simon Parsons and Michael Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

47. Philippe Pasquier, Roberto Flores, and Brahim Chaib-draa. Modeling flexible social commitments and their enforcement. In M.-P. Gleizes, A. Omicini, and F. Zambonelli, editors, *Proceedings of Engineering Societies in the Agents World V, Toulouse, October 2004*, number 3451 in LNAI, pages 153–165. Springer-Verlag, Heidelberg, 2005.
48. M. Pěchouček, V. Mařík, M. Rehák, D. Šišlák, P. Benda, and L. Foltýn. Advanced agent methods in adversarial environment. intermediary deliverable to Air Force Research Laboratory AFRL/EORD research contract (FA8655-04-1-3044), 2004.
49. M. Pěchouček, V. Mařík, D. Šišlák, M. Rehák, J. Lažanský, and J. Tožička. Inaccessibility in multi-agent systems. final report to Air Force Research Laboratory AFRL/EORD research contract (FA8655-02-M-4057), 2004.
50. M. Pěchouček, V. Mařík, and O. Štěpánková. Role of acquaintance models in agent-based production planning systems. In M. Klusch and L. Kerschberg, editors, *Cooperative Information Agents IV - LNAI No. 1860*, pages 179–190, Heidelberg, July 2000. Springer Verlag.
51. M. Pěchouček, V. Mařík, and J. Bárta. A knowledge-based approach to coalition formation. *IEEE Intelligent Systems*, 17(3):17–25, 2002.
52. M. Pěchouček, V. Mařík, and O. Štěpánková. Role of acquaintance models in agent-based production planning systems. In M. Klusch and L. Kerschberg, editors, *Cooperative Information Agents IV - LNAI No. 1860*, pages 179–190, Heidelberg, July 2000. Springer-Verlag, Heidelberg.
53. M. Pěchouček, V. Mařík, and O. Štěpánková. Towards reducing communication traffic in multi-agent systems. *Journal of Applied Systems*, 2(1):152–174, 2001. ISSN 1466-7738.
54. Witold Pedrycz and Fernando Gomide. *An Introduction to Fuzzy Sets : Analysis and Design*. Complex Adaptive Systems. Bradford Book, Cambridge, London, 1998.
55. Don Perugini, Dale Lambert, Leon Sterling, and Adrian Pearce. A distributed agent approach to global transportation scheduling. In *The 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003)*, pages 18–24, Halifax, Canada, 2003.
56. Don Perugini, Dale Lambert, Leon Sterling, and Adrian Pearce. Agent-based global transportation scheduling in military logistics. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1278–1279, Washington, DC, USA, 2004. IEEE Computer Society.
57. Eric A. Posner and Alan O'Neil Sykes. *Optimal war and jus ad bellum*, 2004.
58. Matthew Rabin. Risk aversion and expected-utility theory: A calibration theorem, 2000.
59. Sarvapali Ramchurn, Nicholas Jennings, Carles Sierra, and Lluis Godo. Devising a trust model for multi-agent interactions using confidence and reputation. *Applied Artificial Intelligence*, 18(9-10):833 – 852, 2004.
60. S.D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multiagent systems. *The Knowledge Engineering Review*, 19(1), 2004.
61. M. Rehák, M. Pěchouček, J. Tožička, and D. Šišlák. Using stand-in agents in partially accessible multi-agent environment. In *Proceedings of Engineering Societies in the Agents World V, Toulouse, October 2004*, number 3451 in LNAI, pages 277–291. Springer-Verlag, Heidelberg, 2005.
62. Martin Rehák, Lukáš Foltýn, Michal Pěchouček, and Petr Benda. Trust model for open ubiquitous agent systems. In *Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference*, number PR2416 in IEEE, 2005.
63. Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195. ACM Press, 2001.
64. M. Sierhuis, J. Bradshaw, A. Acquisiti, R. van Hoof, Jeffers R., and A. Uszok. Human-agent teamworks and adjustable autonomy in practice. In *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS - NARA*, Japan, May 2003.
65. Olivier Simonin and Jacques Ferber. Modeling self satisfaction and altruism to handle action selection and reactive cooperation. In *proceedings Supplement SAB 2000, The Sixth International Conference on the Simulation of Adaptive Behavior, FROM ANIMALS TO ANIMATS 6 (Paris, France)*, pages 314–323, 2000.
66. D. Šišlák, M. Rollo, and M. Pěchouček. A-globe: Agent platform with inaccessibility and mobility support. In M. Klusch, S. Ossowski, V. Kashyap, and R. Unland, editors, *Cooperative Information Agents VIII*, number 3191 in LNAI. Springer-Verlag, Heidelberg, September 2004.
67. R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. In *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

68. D. Stauffer. *Introduction to percolation theory*. Taylor and Francis, London and Philadelphia, 1985.
69. The Sun. Java 2 edition 5.0. <http://java.sun.com>, 2004.
70. Sun Microsystems. Java 2 Platform, Micro Edition (J2ME). <http://java.sun.com/j2me/index.jsp>.
71. Sun Microsystems. Improving Serialization Performance with Externalizable. <http://developer.java.sun.com/developer/TechTips/2000/tt0425.html>, November 2002.
72. Niranjan Suri, Marco M. Carvalho, Jeffrey M. Bradshaw, Maggie R. Breedy, Thomas B. Cowin, Paul T. Groth, Raul Saavedra, and Andrzej Uszok. Enforcement of communications policies in software agent systems through mobile code. In *POLICY*, pages 247–250, 2003.
73. K. Sycara, J. Lu, M. Klusch, and S. Widoff. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOID Record*, 28(1):211–246, 1999.
74. Katia Sycara. Multi-agent infrastructure, agent discovery , middle agents for web services and interoperation. In *Mutli-agents systems and applications*, pages 17–49. Springer-Verlag New York, Inc., 2001.
75. Vincent Thomas, Christine Bourjot, Vincent Chevrier, and Didier Desor. Hamelin : A model for collective adaptation based on internal stimuli. In Stefan Schaal, Auke Ijspeert, Aude Billard, Sethu Vijayakumar, John Hallam, and Jean-Arcady Meyer, editors, *From animal to animats 8 - Eighth International Conference on the Simulation of Adaptive Behaviour 2004 - SAB'04, Los Angeles, USA*, pages 425–434, Jul 2004.
76. Steven Willmott, Alan Bundy, John Levine, , and Julian Richardson. An adversarial planning approach to go. In *Proceedings of the Firstst International Conference on Computers and Games*, pages 93–112. Springer-Verlag, LNCS 1558, 1998.
77. Cees Witteveen, Nico Roos, Roman van der Krogt, and Mathijs de Weerd. Diagnosis of single and multi-agent plans. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 805–812, New York, NY, USA, 2005. ACM Press.
78. Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 14–27. ACM Press, 2003.



Autonomous Agents in Air-Traffic Control: Phase 1 Final Report

this is a final project report to the FA8655-04-1-3044-P00001 extension of the FA8655-04-1-3044 contract and it provides technical information about the status of the project work after completion Phase 1 (month 0 – month 6) of the 'Autonomous Agents in Air-Traffic Control' project

Michal Pěchouček^{PI}, David Šišlák,
Dušan Pavlíček, Miroslav Uller

Agent Technology Group, Gerstner Laboratory,
Czech Technical University in Prague

Abstract:

The aim of the FA8655-04-1-3044-P00001 project is to deploy the multi-agent technology for the deconflicted air-traffic control among several autonomous aerial vehicles (manned as well as unmanned). Within the first six month of this research effort technical implementation of the agent based model of the autonomous flight has been carried out. The agent based model of the individual flights and models of interaction have been deployed within AFRL funded **A-globe** multi-agent system.

The preliminary version of the negotiation based deconfliction process have been developed and integrated in the **A-globe**-based model of the individual flight. The system operation has been integrated with freely available, geographical and tactical data sources. This has been done in order to demonstrate the openness of the system to integrate with real data. An additional, web client visualisation and access component has been developed in order to facilitate a multi-user, platform independent use of the system. The results of the first phase of the project has been described in technical detail in this report and are illustrated in the enclosed *.avi video sequence.



Delivery structure

This delivery consists of this report and demonstration disc and contains the elements listed hereafter:

1. Paper and electronic version of this report
2. source codes – all JAVA sources of the ATC system with comments, **A-globe** sources are included,
3. two demonstration scenarios
4. ATC system source codes documentation, demonstration scenarios description
5. *.avi video demonstrating the results.

This delivery is going to be delivered in electronic and paper forms. The final delivery will be available for download from:

`ftp://deconfliction:FA8655-04-1-3044-P00001@agents.felk.cvut.cz`



1 Project Objectives

The specific objectives of the FA8655-04-1-3044-P00001 project have been to provide:

- multi-agent model of individual flight deployed in the **A-globe** multi-agent system,
- 3D visualization component and 2D web-client access component,
- component providing a route plan within times specific way-points avoiding no-flight zones,
- rule-based deconfliction negotiation mechanisms among multiple aerial vehicles.

In the following we provide technical description of the delivered components and explanation how the listed objectives are matched.



2 System Structure Overview

Air Traffic Control (ATC) system is mainly written in JAVA language except real-time visualization component which is written in C++ and is pre-compiled for Windows 32bit operating system. The system can be easily spread on more host computers with different operating systems. The system consists of several components, see Figure 1:

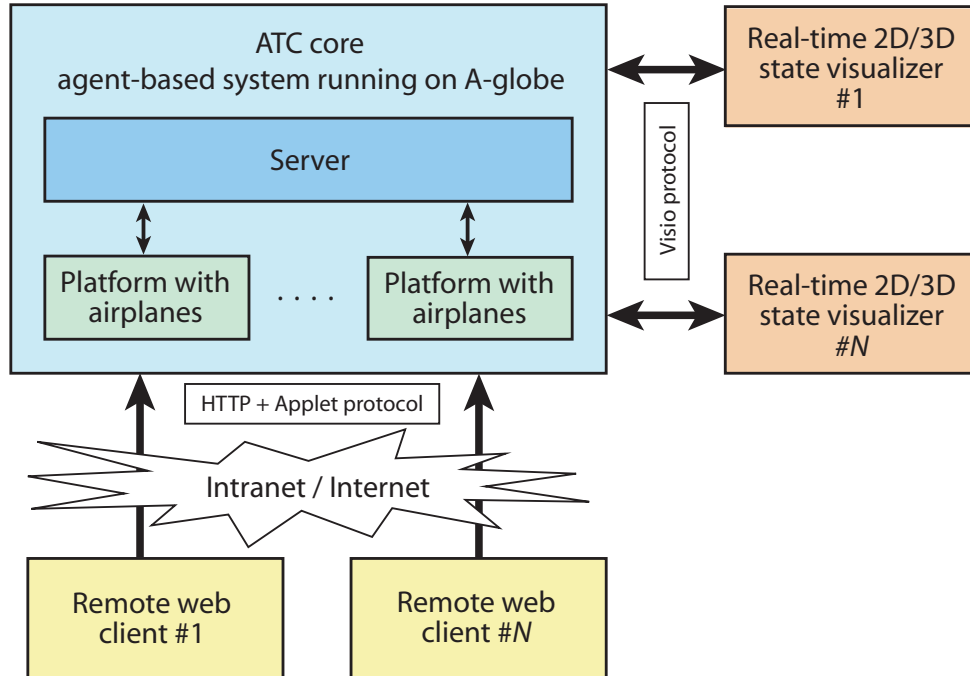


Figure 1: ATC System Structure Overview

- **ATC core** – mandatory component of the system responsible for aircraft simulation and airways planning. All parts of this component are represented by agents which are running on *A-globe* JAVA multi-agent platform [1, 4]. This component also provides interfaces for connection several number of real-time visualization components and remote WEB clients. The architecture of this component is described in the section 3.
- **Real-time 2D/3D visualizer** – optional component that provides real-time overview of the simulation state with all important information in a 3D/2D environment to the user. There can be more visualizers connected to the system at the same time and each can provide information from different area of the simulated environment. Data are transferred between ATC core and a visualization component by the TCP/IP connection using special binary protocol for fast data encoding/decoding. The component architecture and visualization capabilities can be found in the section 5.
- **Remote WEB client** – optional component allowing remote user to connect and interact with the ATC core system. In current version the user can only display information which he needs. The component and protocol are ready for implementation of active user-controlled



action that will have impact to the future simulation state, e.g. the user will be able to define and insert new flight to the system or change mission specification of some aircraft. The client is simply started by the entering URL address of the ATC core system to the internet browser. The client is written in JAVA built up on the JOGL libraries (Java binding for OpenGL [3]) which are used for accessing graphics 3D acceleration. The ATC core system uses Java Web Start application for the client loading and starting. Before a user can use all features of the remote WEB client, he needs to be successfully logged with valid username and password. For minimizing network traffic between the remote client and ATC core system combination of HTTP and special binary protocol is used. Remote client authentication procedure is secured using one-time hashes for password validation. If it is necessary whole data communication can be secured using asymmetric cryptography but it comes with higher processor load requirements. Overview of user interface and data provided via remote client is in the section 5.3.

All system components can run on the same host computer or the system can be spread on more hosts for simultaneous simulation of huge number of aircraft. Components requirements for host machine are specified in the sections 7.1, 7.2 and 7.3.



3 ATC Core

Agent-based ATC core system encapsulates one *server component* (described in section 3.1) and one or more *platform components*, figure 1. Components are running on the **A-globe** JAVA multi-agent platform. The *platform component* is used as a registration unit for starting simulated *aircraft container* (described in section 3.2). Inside on Java Virtual Machine (JVM) only one *platform component* can be started. When ATC system is used for planing/simulation of huge number of aircraft it is highly recommended to use more host computers with own JVMs and *platform containers*. There is no sense for starting more JVMs on the same host it only leads to the higher resource requirements. Whole ATC core system (both *server component* and one *platform component*) can be started inside the same JVM.

In the configuration when whole ATC core system is running on the same host machine, it is highly recommended to start one *platform component* in the same JVM with the *server component*. Appending new containers to the running JVM is allowed by **A-globe** multi-agent platform [4]. In the situation when ATC system is started on the two or more host computers, it is useful to run *server component* separately on the one computer and rest computers has one *platform component* on each of them. The number of running aircraft containers on the more registered platforms is proportionally split between them. By this way the ATC system balances the overall load between all registered computers.

3.1 Server Component

The *server component* of the ATC core system is sole central element of the system. It simulates positions of aircraft and other objects in the simulated world, aircraft hardware, weather conditions, communication ranges given by range of board data transmitters, etc. When the proposed distributed agent system for flying on deconflicted airways will be used to control real aircraft, this *server component* can be removed.

The *server component* of the ATC core system is also responsible for acquiring information about all planes and provides them to both *real-time visualizer* component (section 5) and *remote WEB client* (section 5.3). It works also as a *scenario player* which takes care of creation of new aircraft with a rough plan mission in specific time moment.

The *server component*, figure 2, consists of several agents:

- **Configurator Agent** – handles configuration of the ATC system. It loads initial configurations from the specified configuration files and distributes them to the other agents. The *configurator agent* is also prepared for dynamic configuration changes during the running planing/simulation.
- **Plane Manager Agent** – administrates connected JVM platforms and running aircraft containers, starts new or removes existing planes, gives initial flight mission to the plane. New planes can be started from the graphical user interface or created automatically as specified in the scenario script. The new aircraft container is started on the registered platform where there are minimum running plane containers. This works as a load balancer between connected host computers where the ATC core system is running. Any running aircraft can be removed from the ATC system using this manager.
- **Plane Simulator Agent** – computes true position of the aircraft in the simulated world. It contains all physical models for the all plane types and holds all current flight plans and

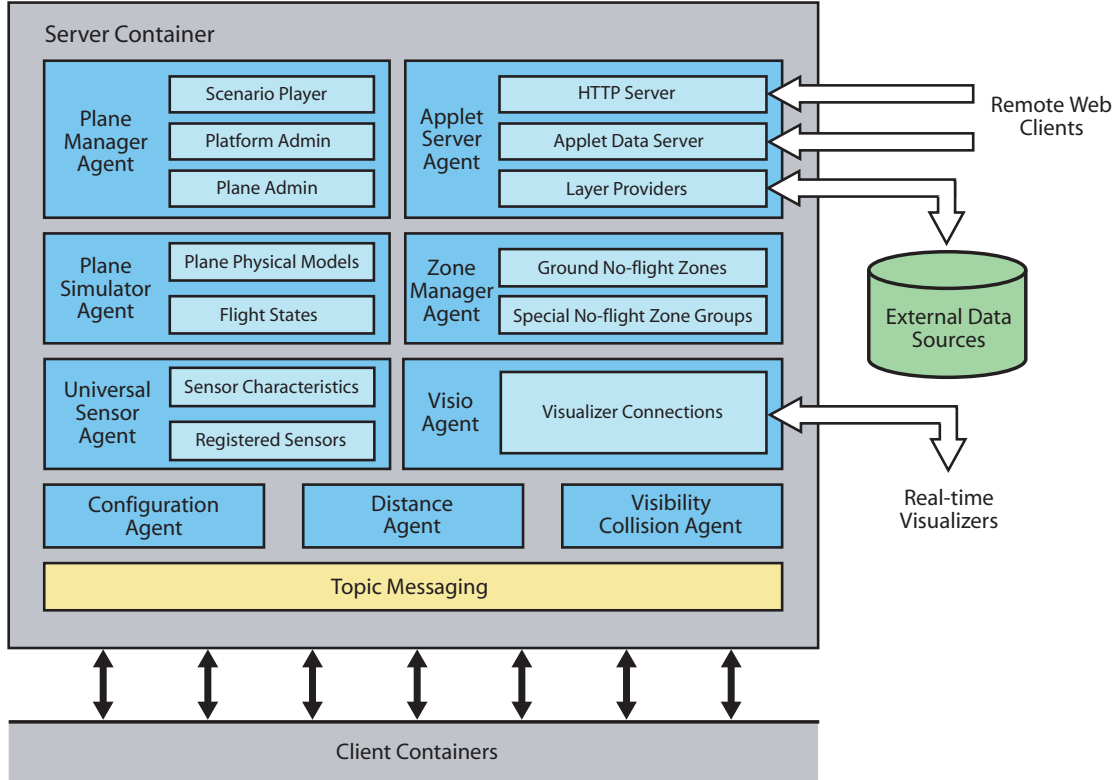


Figure 2: Server component of the ATC Core system

states for running aircraft. When a plane *pilot agent* changes some part of the flight plan, the change is propagated via *plane agent* to this *plane simulator agent* by the difference flight plan update. The details about physical modelling of aircraft can be found in the section 4.1. The agent can be asked for the plane position in current simulation time by the *pilot agent*.

- **Distance Agent** – counts euclidian distances between every pair of existing aircraft using the positions generated by the *plane simulator agent*.
- **Visibility Collision Agent** – prepares *A-globe* visibility updates [4] for controlling communication restrictions between all *A-globe* agent containers. It also detects whether there is physical collision between flying aircraft. If collision is detected the *plane simulator agent* is notified about it. The airplanes that had collision with other object are uncontrollable and they go down to the ground. Falling aircraft can endanger any plane which flies under it.
- **Universal Sensor Agent** – represents all radar sensors on aircraft boards. *Plane agents* can register a specific sensor in it. The *sensor agent* sends radar information to the registered agents depending on the sensor characteristics and aircraft positions and orientations.
- **Zone Manager Agent** – keeps no-flight zones. It transforms any defined no-flight zones to the compressed octant tree. The ground surface is also represented as a special no-flight zone. This allows use of detailed flight plan planning mechanism also to the flight planning without collision with the ground surface. No-flight zones can be dynamically changed during the planning/simulation. Different aircraft can use different no-flight zones.
- **Visio Agent** – is an interface between the JAVA agent system running on *A-globe* and the C++ real-time visualizers. The agent provides bi-directional communication. Commands



thus can be send from the visualizer user interface back to the ATC system. For fast communication there is special binary protocol defined for visualizer connection.

- **Applet Server Agent** – runs HTTP server, Applet Data server and all external data providers. It provides communication interface between ATC agent system and the *remote WEB client* (described in the section 5.3). For allowing access to the ATC system from the networks shielded from the internet by the proxy servers the HTTP server listens on standard TCP/IP port 80 and Applet Data server listens on the port 443 normally used for HTTPS protocol (this port is typically tunnelled in such networks). ATC system generates cache and proxy controls headers for controlling cache content validity. The *Applet server agent* has registered data layer providers which provide external and internal content to the *remote clients*. All integrated data layer contents are described in the section 5.3.1. Currently providers read external content from the local dist store but it can be easily changed to integrate any other on-line data source to the system.

All server agents communicate together using *A-globe topic messaging* described in [4]. All used topics are defined in the project class `atc.simulation.global.TopicConstants` where the description of the topic content can be also found. The communication between *server agents* and platform hosted agents also utilizes *topic messaging* due to its easy usage without any complicated addressing. In the project class `atc.utils.TopicConstants` there are defined all topics used for client-server communication.

3.2 Platform Component with Running Airplane Containers

As described in the section 3, the *platform container* is used as a registration unit for starting simulated aircraft. In the system there can be more *platform components* connected to the server. It is depending on the number of computers used in the configuration. Design of the *platform component* is shown in the Figure 3. In the *platform container* there is only one agent in that container running called *platform agent*. The agent acts as a control bridge between *plane manager agent* and the local JVM where it is running. Through this agent *plane container* can be started or removed. In one *platform component* (inside one JVM) there can run many *plane containers* or there can be no *plane container* at all.

The started *plane container* hosts two agents:

- **Plane Agent** – provides higher-level of plane functions, such as flight plan execution in cooperation with *plane simulator agent*, radar and detector readings, plane configuration and time synchronization ticks. The plane configuration includes aircraft type and its mission specification with rough flight plan (see section 4.3) - time constrained or no constrained waypoints.
- **Pilot Agent** – is main control unit of the simulated aircraft handling negotiation based deconfliction as described in the section 6. It processes notification about new visible object on the radar and tries to communicate with respective *pilot agent* if there is some. It plans detailed flight plan from the initial mission specification given by the *plane manager agent* with respect to the no-flight zones (section 4), described in the section 4.3. It is also responsible for collision detection between its flight plan and the other flight plan part (see section 6).

The plane container agents communicate with the server container via *A-globe topic messaging*. While together they communicate by standard agent communication language (ACL) messages.

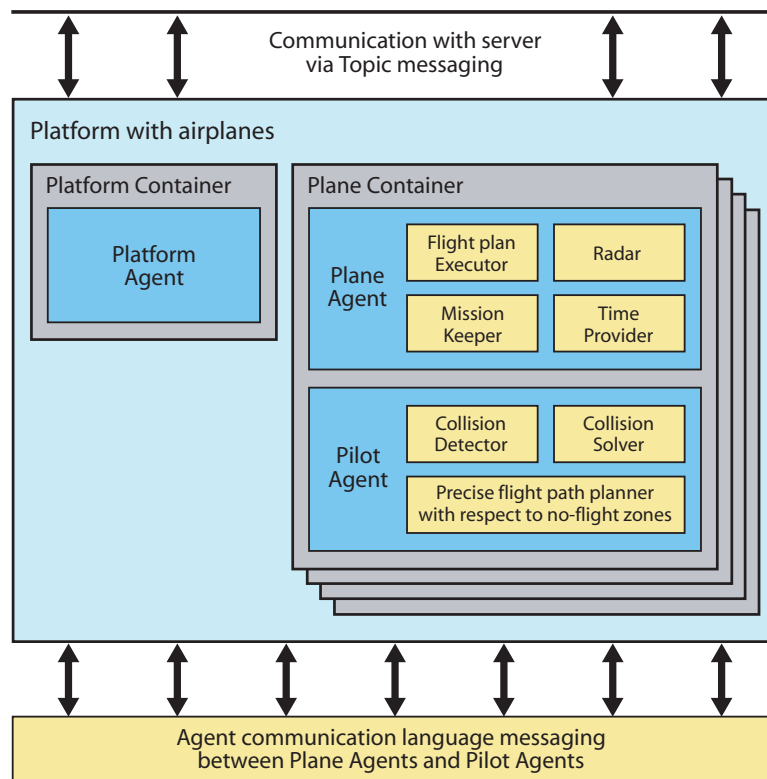


Figure 3: Architecture of the platform with running plane containers



4 Flight Models in \mathcal{A} -globe

In this section we will give a more detailed description of the components of our system which are responsible for the modeling of the flight of airplanes. The flight modeling proceeds in two main phases:

- **planning** – a flight plan is composed for each airplane, describing the trajectory that the plane will follow; and
- **simulation** – the actual flight of airplanes is simulated according to their flight plans.

The airplane model and the structure of flight plans used in our simulation system were designed so that the simulation is very fast and the simulation system is able to plan the flight paths and simulate the flight of a great number (hundreds to thousands) airplanes at once. That is why we use a very simplified physical model of airplanes in our simulation.

4.1 The Airplane Model

In our system, airplanes are modeled as mass points, which move along a previously planned trajectories. The state σ of an airplane in given time t is defined by the following parameters: the center of mass of the airplane \mathbf{P} , direction vector \mathbf{d} , normal vector \mathbf{u} , velocity v and acceleration a . The direction vector \mathbf{d} corresponds to the flight direction. The vector \mathbf{u} , normal vector or *up-vector*, is perpendicular to the direction vector and always aims upwards. Let us denote vector $(\mathbf{d} \times \mathbf{u}) / \|\mathbf{d} \times \mathbf{u}\|$ (an unit vector perpendicular to vectors \mathbf{d} and \mathbf{u}) as \mathbf{w} ; then, the quadruple $(\mathbf{P}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ defines the local coordinate system bound to the current state of the plane, as opposed to the world coordinate system $(\mathbf{O}, \mathbf{x}, \mathbf{y}, \mathbf{z})$, see Figure 4. The plane parallel to vectors \mathbf{d} , \mathbf{w} and passing through \mathbf{P} is referred to as *plane of flight*.

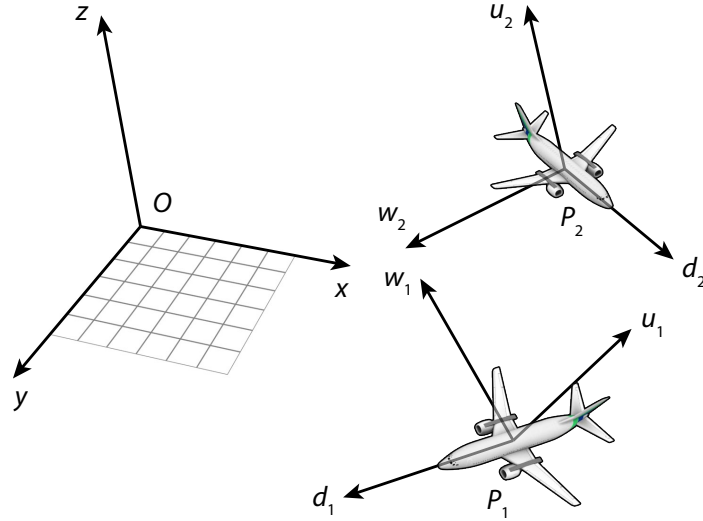


Figure 4: World and local coordinate systems

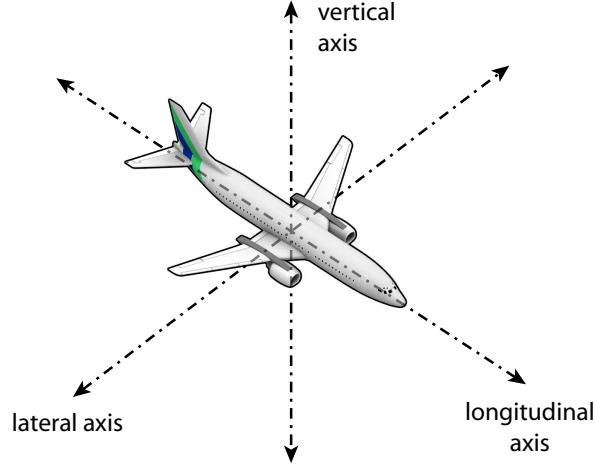
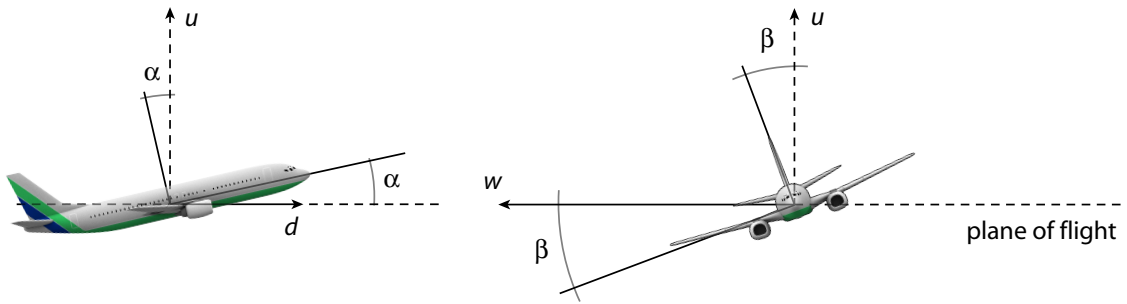


Figure 5: Axes of the airplane

This definition of local coordinate systems of the airplanes allows us to specify the flight plans for the airplanes in a very simple way (the flight plans are described in detail below). However, it is important to note that this coordinate system is linked with the motion of the airplane and that the directions of vectors \mathbf{d} , \mathbf{w} , \mathbf{u} do not necessarily correspond to the directions of longitudinal, lateral and vertical axes of the airplane (see Figure 5). As opposed to simplified mass point model, the real airplane has a specific geometry and shape of wings. For example, the lift and drag forces affect the airplane movement thus the direction of flight of the airplane is not identical to its main (longitudinal) axis, but there is nonzero angle α between the (frequently referred to as the angle of attack). Also, during turns, the airplane is affected also by the centrifugal force, which forces the airplane to turn around its longitudinal axis; the angle between the plane of flight and the plane of the wings of the airplane is called bank angle β . See Figure 6.

Figure 6: Angle of attack (α) and bank (β) angles

The five main forces, which affect the movement of an airplane during the flight, can be expressed by the following simplified formulas:

- gravity $\mathbf{G} = m\mathbf{g}$,
- lift $\mathbf{L} = \frac{1}{2}C_L A \rho^2$,



- drag $\mathbf{D} = \frac{1}{2}C_D A \rho^2$, where $C_D = C_{D_{min}} + \frac{C_L^2}{\pi \cdot ar \cdot 0.75}$,
- thrust \mathbf{T} , and
- centrifugal force $\mathbf{C} = \frac{mv^2}{R}$, present during turning.

The meaning of the parameters mentioned above is as follows: m is plane weight, \mathbf{g} is the acceleration of gravity, C_L is the coefficient of lift, C_D is coefficient of drag, A is the area of wings, ar is the aspect ratio of wings, ρ is the air density, v is the airplane velocity and R is a radius of turning. Our airplane model is very simple; we do not use the aforementioned formulas as the equations of motion to describe the Newtonian airplane dynamics, but we use them only to determine the magnitudes and directions of the forces affecting the airplane. This information is then utilized for getting the pitch and roll angles of the plane and for computing the thrust force. The pitch (angle between the longitudinal axis of an airplane and xy plane) and roll (angle between the lateral axis of an airplane and xy plane) angles are used for the simulation of the radars (since the area of the airplane profile visible from the ground depends on the banking of the airplane). From the magnitude and direction of thrust we can estimate e.g. the fuel consumption.

Our model neglects certain more complicated aspects of the airplane geometry, such as flaps or landing gear. Furthermore, in the current version of the simulation system, we assume the airplane weight to be constant during the entire flight – we do not consider the decrease of the weight due to the gradual fuel consumption. We also do not (yet) consider the impact of wind on the velocity and direction of the airplane movement; we assume that the plane performs such maneuvers, that it always follows its planned trajectory and moves at the defined velocity. As for now, we utilize this kind of information only to specify the direction of the airplanes during takeoff and landing, in the way that the airplanes land and take off in the direction against the direction of wind.

4.2 The Structure of Flight Plan

Basically, the flight plan describes a trajectory, which the plane will follow during its flight. The flight plan is represented by two-level data structure, whose principal building blocks are waypoints, segments and elements. Waypoints and segments represent the high-level description of a flight plan (rough flight plan); elements represent the low-level description (detailed flight plan).

- A **waypoint** contains the information about a particular location (its position in the world coordinates and possibly some textual description) and time, when the plane is to fly through the waypoint. The temporal data associated with the waypoint include the time of arrival t and the time tolerance δ . During the planning phase, the flight paths are created in such a way that the airplane flies through - if possible - the waypoint in the time interval $(t - \delta, t + \delta)$. It is possible to define an “exact” waypoint with a zero time tolerance, or a waypoint with no time limitations at all.
- A **segment** represents a part of a flight plan between two successive waypoints. Each segment consists of a sequence of flight plan elements. One segment typically contains two to six elements.
- An **element** is the most basic building block of a flight plan. It represents a part of an airplane trajectory, which is described by a simple curve (like line, circular arc, spiral...). Each element describes the flight path of the airplane relative to some *initial state* of the airplane, which we will denote σ_0 . The element contains the following information describing the flight path: the description of the curve (the type of the curve, possibly with some



parameters) and the duration δ of flight for this element. The duration parameter is present for all types of elements. The state, which the airplane reaches when it will fly off the whole element (i.e. after the time δ has elapsed) we will denote as *final state* σ_1 (with respect to the element). If we use function notation, then the element can be considered as the function $\sigma(t)$ of state over time (more exactly, duration). Then, it holds that $\sigma(0) = \sigma_0$ and $\sigma(\delta) = \sigma_1$. There are four main types of flight plan elements (see Figure 7):

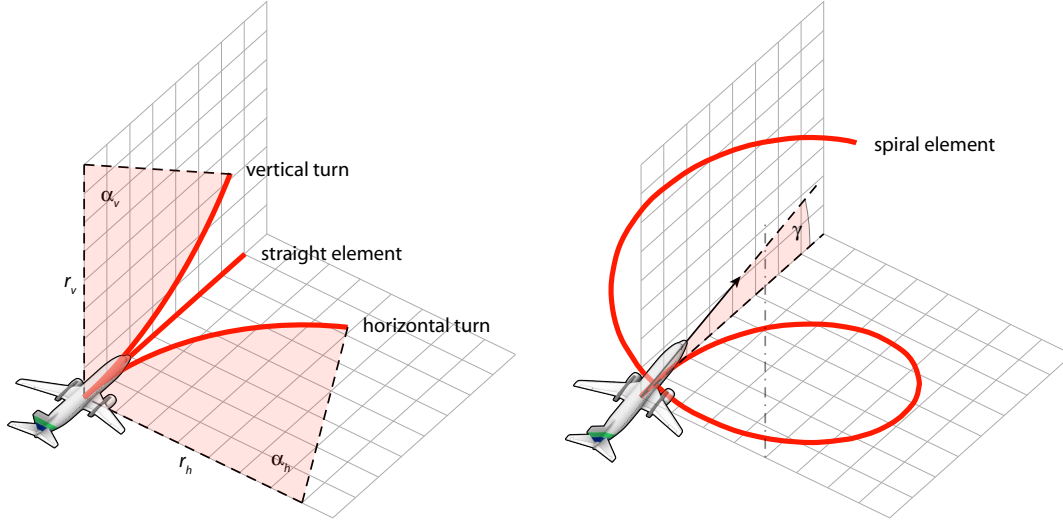


Figure 7: Flight plan element types

1. **Straight element**, with two parameters: duration and acceleration. The airplane will simply fly with the constant acceleration for the specified time in the direction given by its initial state σ_0 (i.e. in the direction of the direction vector \mathbf{d}). The final velocity (at the end of the element) is determined by the initial velocity, duration and acceleration.
2. **Horizontal or vertical turn elements** are represented by circular arcs. Horizontal turns are performed in the plane of flight and correspond to a common notion of making a turn. Vertical turns are performed in the plane described by the initial airplane state vectors \mathbf{d} and \mathbf{u} , perpendicular to the plane of flight; they are used to represent parts of flight path, where the plane e.g. changes its motion from a straight flight to an ascent/descent or vice-versa. The parameters of both types of turns are the duration and the turning radius. During the turning the airplane's velocity remains constant.
3. **Ascending or descending spiral** (aka world horizontal turn). Represented by a part of a spiral, with an axis parallel to the z axis of the world coordinate system. The parameters of the spiral are the duration and the turning radius. The rate of climb, or descent respectively, is derived from the direction vector \mathbf{d} of the initial state σ_0 ; its parameter γ (see Figure 7) is the angle between the vector \mathbf{d} and its projection to the plane xy .
4. **Warp element**. A special kind of element, which does not contain any trajectory; its sole purpose is an immediate change of state parameters – position, direction and velocity. It is used mainly for the initial setup of the airplane state at the beginning of a flight plan (the first element of every flight plan is always the warp element). This element has zero duration.



The more complicated flight plans are created by joining various types of flight plan elements into a sequence. An example of such flight plan can be seen on Figure 8. The initial state of each element can be obtained from the final state of the previous element. The warp element represents an exception, since it sets all state variables of an airplane to new values regardless of its previous state.

For the sake of simpler implementation of some algorithms and operations over flight plans, the warp elements are used not only for the initial setting of state parameters at the beginning of the flight plans, but we place them at the start of each segment. Aside from simplifying of certain operations with segments, this measure effectively reduces the influence of possible cumulative rounding errors, which may accumulate during the computations of the final states of the elements.

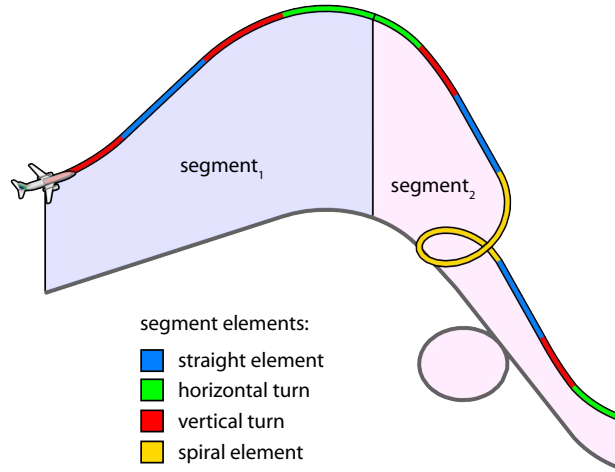


Figure 8: Flight plan, segments, elements

4.3 Flight Path Planning

The planning of the flight path of an airplane is performed by the planner, a component of the pilot agent, which resides on the container associated to the airplane. The result of the planning is the flight plan, which was described in detail above. The inputs to the planner are the rough flight plan for the airplane (represented by the list of waypoints, which the airplane has to visit) and the velocity, which the airplane should have at the beginning of its route. In addition, the planner may also consider the information about zones, which the plane must not enter during its flight, and plan the path to avoid them.

Aside from the planning of new flight paths, another important function of the planner is the so-called *replanning*. The replanning changes a part of an existing flight plan while keeping the rest of the plan intact. We use the replanning e.g. in collision detection and avoiding the collision zone (see 6.2). Also, replanning is used to alter the flight plan to avoid the no-flight zones (we describe them in more detail later). The replanning is usually performed by means of inserting of special waypoints into a particular segment or segments. The insertion or insertions will cause the splitting of the replanned segment into two or more new segments; these segments are to be planned again, thus creating a replanned flight path. Aside from the ordinary waypoints, used for the definition of the important navigational points for planning the flight path of an airplane, there are two more



types of waypoints. The *solver* waypoints are used during the collision detection and avoidance and the auxiliary *temporary* waypoints are used to define the path for avoiding the no-flight zones.

The planning of a flight path proceeds in three main phases. In the first phase (path planning), the planner generates an initial flight plan, which passes through all input waypoints. If there are any time constraints associated with the waypoints, they are ignored in this phase. The planned path is created to be as short as possible. In the next phase (time planning), certain parameters of the segments generated in previous step are adjusted in such way that the modified flight plan satisfies (if possible) all time constraints. Finally, if there are defined any forbidden (no-flight) zones for the airplane and the flight plan generated in the previous phases collides with any of them, the plan is in the third phase (avoiding the no-flight zones) modified to avoid them.

4.3.1 The Path Planning

In this phase, the planner generates a detailed flight plan in such way that the flight path will correctly pass through all the waypoints. The temporal data associated with each waypoint are not yet taken into account. For each couple of successive waypoints, a segment is generated. The segment is empty, containing no elements. A segment represents the smallest part of flight plan, which can be planned independently on the other parts of flight plan. Each segment has several parameters, serving as inputs for planning algorithm (which will fill the initially empty segment with elements). These parameters are the start and end points of the segments with the tangents to the flight plan in these points. The tangents are calculated from the input set of waypoints and chosen in such a way that the tangent in the end point of some segment and the tangent in the start point of the next segment will point in the same direction (this property is called geometric continuity or G^1 ; the planning algorithm, which will be described below, assures that the same condition holds also for the elements of the segment; therefore, the planned flight path is always G^1 continuous).

With these parameters (endpoints and tangents), we can perform the planning. The output of the planner will be in the form of the sequence of flight plan elements. To facilitate the planning algorithms, the planner uses a representation of the elements different from the one described above. Instead of specifying the elements relative to some previous state (which is useful i.e. during the flight simulation performed by plane simulator), we use a representation containing all parameters necessary to determine the exact location and shape of the element. For example, a straight element is defined by the start and end points of the line and the acceleration.

The new flight plans are created by using three types of elements: straight elements, turns (horizontal or vertical) and spirals. The fourth type of element, the warp element, is used only at the beginning of each segment for the purposes of setting the initial velocity of the segments in the time planning phase; we do not use them during the path planning. For the purposes of planning, the horizontal turns are used in the cases, where it is necessary to change the direction of flight (more exactly, the heading of the airplane) in some place. The vertical turns are used in places where it is necessary to change the pitch of the airplane (i.e. when the airplane starts climbing). The spirals are used mainly in the situations, when it is necessary to rapidly change the flight level of an airplane (its altitude). We distinguish between ascending spirals (with positive elevation), descending spirals (with negative elevation) and waiting loops (with zero elevation). Waiting loops represent the situation, when the airplane flies in circles around a fixed point.

It is obvious that the parameters of turns and spirals cannot be chosen completely arbitrarily. For example, the airplane cannot turn on the spot and also cannot climb or descend under arbitrarily steep angle (the last statement is not entirely true, since there are some airplanes that



fit into this category, but they are rare exceptions). Therefore, some limitations must be considered during the planning, such as that the radius of turns and spirals cannot be less than certain minimum (this parameter depends on the type of airplane and is defined as the radius of smallest turn, which the airplane is able to perform when flying at maximum velocity). Another limitation is that the pitch angle must not be greater than certain limit.

We will now focus on the description of the method which we use for generating a flight plan for a given segment. In the most general case, the segment which is to be planned connects two points \mathbf{A} , \mathbf{B} with different altitudes, whose respective tangents \mathbf{a} , \mathbf{b} have different directions and nonzero pitches (i.e. angles between the tangents and their projections into xy plane).

The method is based on the notion that if we have some segment configuration $(\mathbf{A}, \mathbf{B}, \mathbf{a}, \mathbf{b})$ for which is difficult to create a flight plan, it is often possible to reduce this case a simpler one by inserting suitable elements to the start and end of that segment. Let \mathbf{A}' be the end point of the element inserted to the start of the segment and \mathbf{B}' the start point of the element inserted to the end of the segment (and \mathbf{a}' , \mathbf{b}' the respective tangents). These parameters define a new segment configuration $(\mathbf{A}', \mathbf{B}', \mathbf{a}', \mathbf{b}')$, possibly simpler to plan.

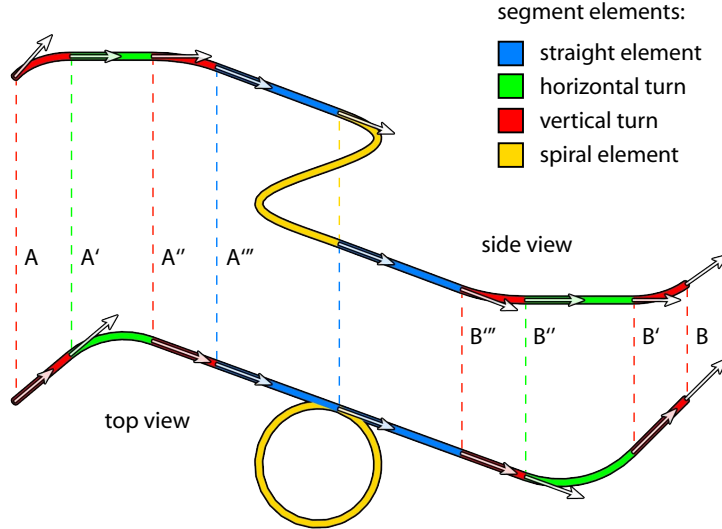


Figure 9: General segment planning

We will show how we can perform the planning by a series of such reductions (see Figure 9). Consider the most general configuration $(\mathbf{A}, \mathbf{B}, \mathbf{a}, \mathbf{b})$ mentioned above. As a first step, we may reduce this case to simpler case, in which the tangents at the start and the end of a new segment will have zero pitch (will be parallel to the plane xy). This can be accomplished by inserting suitable vertical turns to the start and end of the original segment. This reduction yields a new internal segment $(\mathbf{A}', \mathbf{B}', \mathbf{a}', \mathbf{b}')$, whose tangents \mathbf{a}' and \mathbf{b}' have zero pitch. This segment can be further reduced by inserting suitable horizontal turns to its start and end to segment $(\mathbf{A}'', \mathbf{B}'', \mathbf{a}'', \mathbf{b}'')$ such that the tangents \mathbf{a}'' , \mathbf{b}'' have the same direction and this direction is also parallel with the projection of the line $\mathbf{A}''\mathbf{B}''$ to the xy plane. By doing this, the initially three-dimensional planning problem is reduced to two-dimensional one, since the points \mathbf{A}'' and \mathbf{B}'' along with the tangents \mathbf{a}'' and \mathbf{b}'' all lie in the same plane and this plane is perpendicular to the plane xy . This segment can be further reduced by inserting suitable vertical turns to the segment $(\mathbf{A}''', \mathbf{B}''', \mathbf{a}''', \mathbf{b}''')$ such that the tangents \mathbf{a}''' and \mathbf{b}''' and line $\mathbf{A}'''\mathbf{B}'''$ are mutually parallel. The segment $\mathbf{A}'''\mathbf{B}'''$ can be



then easily planned by single straight element connecting the points \mathbf{A}''' and \mathbf{B}''' .

However, it is possible that the pitch of a line $\mathbf{A}'''\mathbf{B}'''$ will be greater than maximum allowed pitch angle for the airplane we are generating plan for. Should this case occur, we can solve it i.e. by dividing the line segment $\mathbf{A}'''\mathbf{B}'''$ in the middle into two line segments and by inserting a climbing (or descending) spiral between these two segments (see Figure 10).

4.3.2 The Time Planning

The flight plan, which is the output of the first phase of planning, constitutes only the initial sketch of the flight route of an airplane; it does not take into account the time relations along it and it ignores all the temporal data assigned to the waypoints. In the next phase of planning, the previously planned segments are adjusted in the way so the resulting flight plan conforms to the time constraints. That is, that the time of flight through each segment would correspond to the time constraints defined in the start and end waypoint belonging to the segment. Basically, we can affect the time which it takes to fly through a segment by three ways:

1. by setting the velocity, which the plane has at the start of the segment,
2. by setting the acceleration parameter of the elements which allows to change the velocity (i.e. of straight elements), or
3. by changing the length of the segment.

The first two ways are obvious, the third requires more detailed explanation. As it was already described above, all segments are planned by following always the same procedure: the original "general" segment is reduced by successive insertions of various types of elements to a segment which is "simpler to plan". After several reductions, we obtain from original segment $(\mathbf{A}, \mathbf{B}, \mathbf{a}, \mathbf{b})$ a simpler segment $(\mathbf{A}'', \mathbf{B}'', \mathbf{a}'', \mathbf{b}'')$, where the projection of the line $\mathbf{A}''\mathbf{B}''$ into the plane xy will have the same direction as both tangents $\mathbf{a}'', \mathbf{b}''$ (see Figure 10).

We then could plan this kind of segment either by a simple sequence vertical turn - straight element - vertical turn with total length of l_1 (but only in the cases when the altitude difference between \mathbf{A}'' and \mathbf{B}'' is relatively small so the straight element will have pitch smaller or equal than maximum pitch for the particular airplane), or in more complicated way by using a spiral element, with total segment length of l_2 (see Figure 10). The latter method of planning (which uses the spiral) allows for the arbitrary stretching the segment over the length l_2 by adjusting the parameters of the spiral and vertical turns. In summary, all segments can be planned to have length greater or equal to l_2 . Furthermore, in the cases where it is possible, the segment can be planned without the use of the spiral, with the length of l_1 .

For now, our time-planning algorithm does not take into account the possibility to alter the airplane velocity when flying over the straight elements of the flight plan. We use the following simplification: the airplane's velocity is constant in the scope of each segment, the acceleration is always zero. In the points where two segments meet (the waypoints), we allow a step change of airplane's velocity. We therefore perform the time planning only by setting the initial velocity at the start of the segments and by adjusting the length of the segments.

The actual time-planning algorithm is very simple, one-pass algorithm, which adjusts the time relations of the segments starting from the first segment and gradually proceeds to the next segments. The algorithm never backtracks; once the segment has been planned, the plan will never

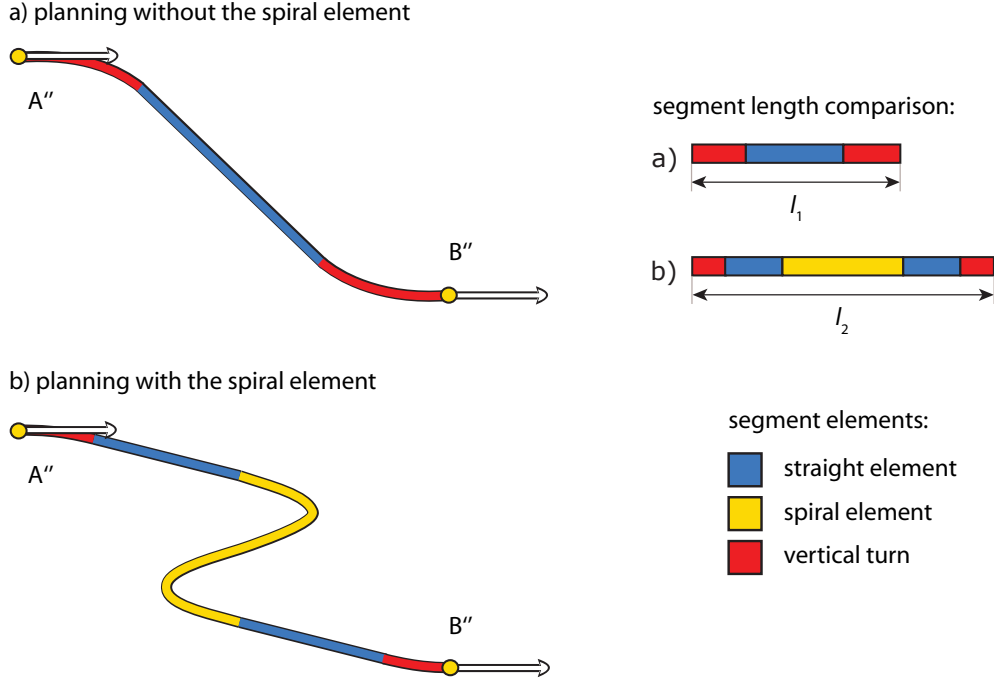


Figure 10: An example of planning a segment of type (A'', B'', a'', b'') . The segment can be planned either by two vertical turns and a straight element or by two vertical turns, two straight elements and a spiral.

change again (speaking in the terms of time planning). The segments are planned in the way that a lower bound of the time allowed for each segment is considered first; the airplane will fly the segment as fast as possible. Since this algorithm in its current form constitutes a radical simplification of the problem, it will be subject to further development and improvement.

4.3.3 Avoiding The No-Flight Zones

One of most important extensions to our flight path planner is the capability to plan the flight path with respect to so-called no-flight zones. By no-flight zones we mean the areas, where the normal flight of airplanes is not possible or permitted for various reasons (e.g. the areas around the nuclear power plants or military installations). The natural terrain obstacles, such as the tops of mountains, can be also considered as a kind of natural no-flight zones. Some types of no-flight zones (such as natural no-flight zones) are obligatory to all types of airplanes; some types are to be considered only by certain types of airplanes. For example, the area around a military base is typically a zone of no flight for all airplanes except for the military airplanes that have a permission to land at the base. For this reason, it is useful to define, in addition to single no-flight zones, also groups of no-flight zones obligatory for certain types of airplanes.

After finding the flight path passing through the input set of waypoints, the planner tests all segments for possible collisions with the no-flight zones. The waypoints and no-flight zones pertaining to a given airplane are always defined that there is no collision between them, so for a segment always holds that both its endpoints lie outside of the defined no-flight zones. In the



case of a segment intersecting a no-flight zone, such a segment is replaced by a flight path, which bypasses all colliding no-flight zones. The endpoints and corresponding tangents of the bypassing flight path will be equal to the endpoints and tangents of the original segment. We construct the bypassing flight path by inserting one or more new waypoints between the endpoints of the original segment and by planning a flight path through these waypoints. Therefore, the flight path to replace the original colliding segment will be composed of two or more segments (see Figure 11).

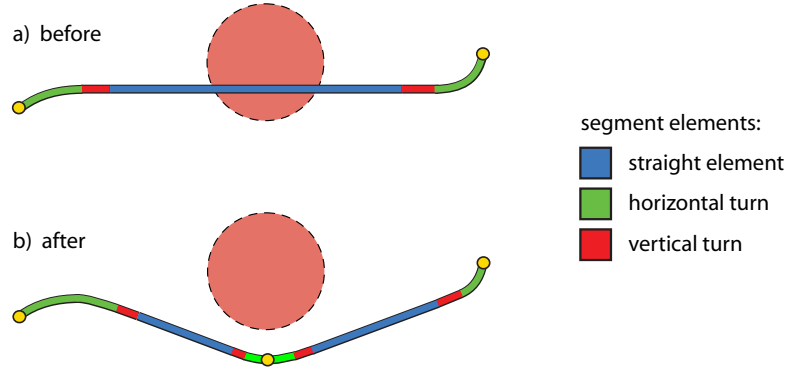


Figure 11: An example of the avoiding the no-flight zone. Seen from top view. The segment collides with a no-flight zone (top picture). The zone can be avoided by inserting another waypoint into the segment and replanning the flight plan (bottom picture).

The problem of the finding of collision-free flight path is therefore equivalent to the finding of one or more "bypassing" waypoints, which will determine the new flight path. The description of an algorithm, which we use to find these waypoints, will be given below, along with the brief description of relevant data structures.

The idea central to our approach can be formulated as follows: the entire area, in which we perform the planning and simulation of a flight of an airplane (or airplanes), is divided by uniform grid to a set of identical cubic (or cuboidal) cells. We then mark all cells, which are intersected by some no-flight zone, as full; the other cells are marked as empty. Let's assume that the waypoints are always placed sufficiently far from the no-flight zones, so even after the "rasterization" of them by the grid neither of them will lie in full cell of the grid. By the rasterization we mean the result of the conversion of no-flight zones to the set of full grid cells; obviously, the no-flight zone is always a geometric subset of its rasterization.

Let's consider a segment, colliding with some no-flight zone or zones. The segment at certain point intersects some full grid cell, although its endpoints lie by definition in empty cells. Our task is to find such a path that will connect both endpoints of the segment, while passing only through empty grid cells. This problem is equivalent to the problem of finding a sequence of empty grid cells, which will form continuous "tunnel" connecting given start and end grid cell. But then, the start and end cells are simply the cells containing the start and end point of the segment, and the cell sequence forming the "tunnel" can be found e.g. by employing the well-known A* algorithm (see the Figure 12). The centers of these cells can then be used as the "bypassing" waypoints, which will define the segments of the new collision-free path.

The algorithm which we have actually implemented is somewhat more complicated than the one described above, but it follows the same basic idea. The main difference is that instead of uniform grid, we utilize an octal tree. This data structure is slightly more difficult to traverse,

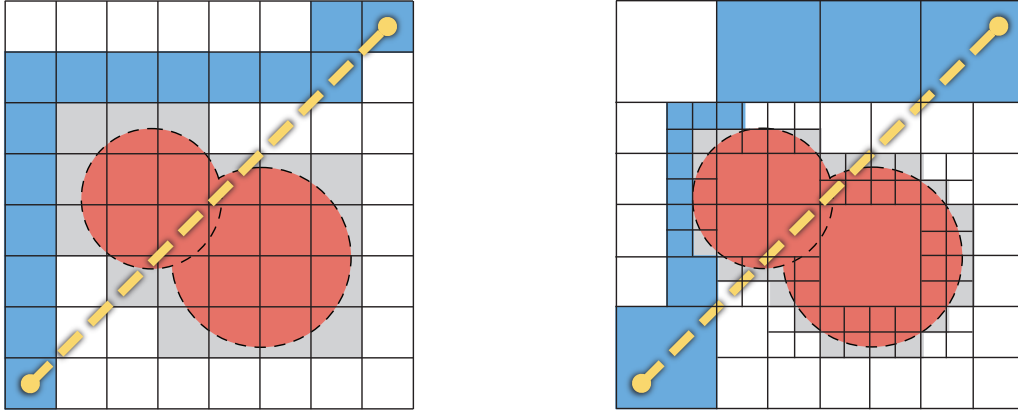


Figure 12: Finding the "tunnel" connecting the grid cells containing the start and end points of the segment. Left picture shows the subdivision by the uniform grid, right picture shows the subdivision by the octree. For the sake of clarity, the examples are shown in 2D.

but it requires much less memory than uniform grid. We represent the no-flight zones as spheres (specified by their center and radius); the octal tree is constructed by successively inserting the spheres into initially empty tree. In the current version of the planner, we utilize only this type of representation of no-flight zones. The no-flight zones with more complicated geometry can be approximated by the union of several spheres. However, the described method can use no-flight zones of arbitrary geometry.

The sequence of the tree cells, found by the A* algorithm, forms a "tunnel" containing the flight path which we seek. In the simplified version of the planning algorithm, we have generated the rough flight path bypassing the no-flight zones as the polyline connecting the centers of the neighboring cells of the "tunnel". Nevertheless, the flight plans generated by this approach were not very optimal and the airplanes often took a route to avoid no-flight zones, which was unnecessarily long. Therefore, the current algorithm uses more sophisticated approach: we find the waypoints of bypassing flight path by running a second phase of A* algorithm (running only over the tunnel cells). The output of this algorithm is also a polyline, but its vertices does not necessarily lie in the center of tunnel cells. The vertices of the polylines found by either approach correspond to the "bypassing" waypoints with no time constraints.

The example of planning of flight paths with respect to no-flight zones can be seen on Figure 25. The flight plans are shown as thick polylines of various colors; the no-flight zones are shown as red, semi-transparent circles.

4.4 Flight Simulation

The simulation of the flight of airplanes is performed by the plane simulation agent, which resides on a server. This agent maintains the information about the state of all airplanes present in the system, along with their flight plans. The flight of airplanes is simulated by the evaluation of their flight plans over the time. The simulation agent is able to simulate the flight of many airplanes at once.



The function of the plane simulator can be well explained on the following analogy. Imagine the flight plan as a movie, which will show the flight of the airplane from the beginning to the end of its flight route. The plane simulator can be then likened to some sort of movie player and the execution of the flight plan by the plane simulator to the playback of the movie on the player. However, there are some notable differences between our plane simulator and ordinary VCR. Most importantly, the plane simulator is able to simulate many flights at once. Also, the flight paths of individual airplanes are not fixed after their first planning, but they can change in future (for example, if two flight plans were set on a collision course, the airplanes will change their flight plans to avoid the collision).

The simulation agent uses the notion of *global simulation time*. It is the time of a global clock, called simulation clock, running on the simulation agent. This time serves as a reference frame for all time information stored in the flight plans. When a new airplane is created, a free "plane slot" is allocated in the plane simulator for its flight plan and state information. After the simulator receives a new flight plan for the plane, it starts executing ("playing") the plan. Every flight plan has defined its own initial global simulation time t_0 (which is equal to the time when the simulator had received this plan) and all time information contained in the flight plan is considered relative to it.

Just as the VCR plays the movie by showing the individual frames every 25-th of a second, the plane simulator executes the flight plans in discrete time steps. By default, the interval between two such "frames" (i.e. updates of the state information of the planes present in the system) shown by the plane simulator is 100 milliseconds.

The simulator maintains up-to-date information about the states of all airplanes currently present in the system. The state information stored in the data structures of the simulator always contains the real state of the airplanes at the moment of the last update. Let's say that the global simulation time of the last update is t and the state of each airplane $\sigma(t)$ describes its position, direction, up-vector, velocity and acceleration. For each plane, the plane simulator also knows its current position in the corresponding flight plan. By the position in the plan, we mean the current segment, the current element and the relative time from the start of the element.

When the next update occurs, the global simulation time will be $t' = t + \Delta t$, where Δt is the time between two successive updates. The state information and the positions of the airplanes in their respective flight plans has to be updated to new values. The state of each airplane is updated by executing the portion of its flight plan corresponding to the time interval between the global times t and t' . During the executions of flight plans, the positions of the airplanes in the flight plans are adjusted accordingly. When the update ends, the new position of each airplane will possibly refer to a new segment, element and/or relative position in the element. The segments and elements, which were finished during the update, are removed from the flight plans. If during the execution of the flight plan of some airplane the end of plan is encountered, the airplane is removed from the simulation system.

The time value Δt may be, but need not to be, equal to the real time passed between two updates (in that case, Δt is equal to the update interval); it even does not have to be constant during the simulation. By changing the Δt value, it is possible to change simulation speed; the simulation can be sped up, slowed down or even stopped. Our simulator allows to perform the simulation with speeds varying up to ten times of the normal simulation speed.

It is also possible to change the flight plan of a plane during its execution, for the reason of changing a current course during the flight. For obvious reasons, only the parts of the plan, which were not yet simulated (the "future" parts of the plan), can be changed. Let's say we want to change a flight plan (i.e. replan it) and the part to be changed (by replacing it with a new plan)



is specified by some element marking its start. The replanning will be performed in the way that the portion of the original plan from the marker element to the end of the plan will be replaced by the new plan.

Although our simulation system is quite complex, it is still in development phase. We are aware that there are many aspects, which will need further development and improvement. The most important of these are the following two:

- The physical model of an airplane. For now, the physical model used in our simulation is sort of imprecise and incomplete. There are some interesting aspects of flight, which we do not consider in the current version of the simulation system, including: the impact of the forces affecting the airplane to the fuel consumption, the decrease of the airplane weight due to the gradual fuel consumption and the influence of wind force on the airplane flight.
- The time planning algorithm. The current time planning algorithm is a very simple one; it is fast (in the terms of performance), but the planned flight path is not necessarily the optimal one. A remarkable disadvantage of this algorithm is that the airplane velocity does not change smoothly over the entire flight plan, but that it is constant in the individual segments and discontinuous at points where two segments meet.



5 Flight visualisation in 3D and 2D

From the very beginning of the project it was obvious that a simulation of this kind will ask for a sophisticated visualization component (visio) that will allow the user to quickly and easily overview the entire simulation in a natural 3D environment, to efficiently navigate through it, but also to provide the user with all the important data and information at the same time. Therefore, a combination of 2D and 3D visual engine was developed.

5.1 System Architecture

As in several previous projects, an open-source 3D game engine CrystalSpace [2] was used as a platform on top of which the visualization component is built. For efficiency and performance reasons, the entire visio is written in C++ and it internally utilizes the OpenGL graphics engine.

Since **A-globe**, on the other hand, is a Java-based simulation system and both **A-globe** and the visio are two completely independent modules, a binary network communication protocol had to be defined and implemented to allow the data exchange between the two.

As a result, several independent visios can be connected to the simulation system at the same time and the user can observe a different part of the 3D scene in each of them. Needless to say that it is possible – and even advisable – to run the visio on a dedicated computer, completely separate from the **A-globe** simulation system itself, allowing for better performance of each of the modules.

It was already mentioned that the network communication protocol is binary-coded as it has been designed with speed and efficiency in mind. The communication between **A-globe** and the visio(s) is duplex: by sending messages to the visio, **A-globe** tells the visio that an airplane has been spawned, destroyed, it has changed its position etc. The visio, on the other hand, sends messages to the simulation system e.g. when it requests specific information about a certain airplane upon user's demand, such as its flight plans. In response to this request, the simulation system selectively sends the required data. This behavior was introduced to reduce the amount of data that need to be sent over the network – instead of sending all the data continuously even if they are not needed.

In the visio, each of the currently existing airplanes (i.e. containers in the terms of **A-globe** architecture) has its own internal message queue (buffer) where all the incoming messages are stored upon arrival and they are processed as soon as possible. Ideally, this would happen immediately. But in case the visio is temporarily or permanently overloaded, the messages cannot be processed fast enough and they could eventually flood the system. To prevent such a situation, old messages of certain type can be discarded once an updated message of the same type arrives. For example, in case of airplane position updates, we only need to know the most recent position and all the previous updates that we didn't have time to process, are not important anymore. Therefore, those messages can be removed from the message queue, effectively reducing the amount of relevant messages to be processed once the visio has the time to do so.

It should be pointed out that due to the communication mechanism described above, the visio must be started before the **A-globe** simulation begins. Obviously, if the visio was executed after the airplanes were created in the simulation system, the visio would never know that these airplanes exist and it would ignore all subsequent messages related to these airplanes. Also, in the beginning of the simulation process, **A-globe** searches for all visios that may be listening by sending a broadcast query. Subsequently, it communicates only with the visios that replied to the



initial query. Therefore, all visios that are executed later in the process, remain invisible to the *A-globe* simulation system.

5.2 User Interface Overview

The visio provides two main display modes: a two-dimensional (2D) view (see Figure 13) and a three-dimensional (3D) view (Figure 17). They both work with the same underlying data, only the way of depicting them differs. The user can switch between these two modes easily by pressing the M button.

The 2D mode provides a radar-like top view of the scene. The user can zoom in and out as needed by pressing the UP and DOWN arrow keys and scroll/pan the view by dragging the mouse with the right mouse button pressed.

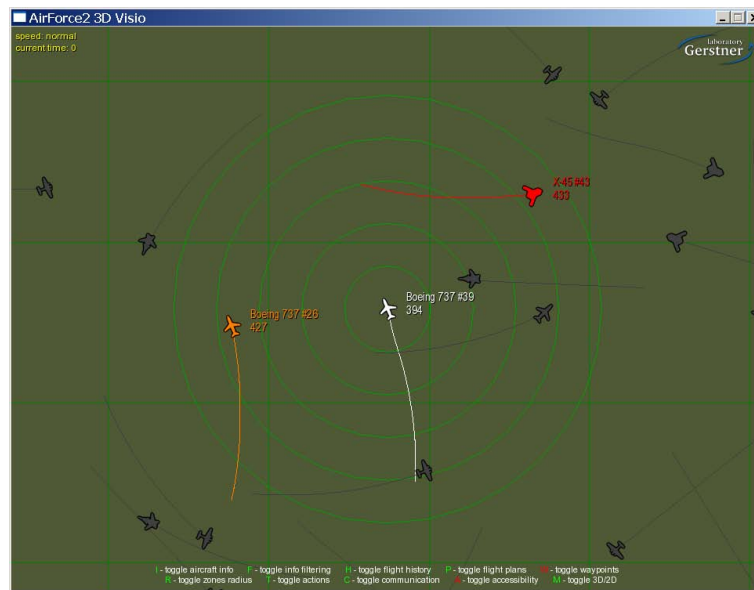


Figure 13: 2D view

Airplanes present in the system are displayed as 2D icons that visualize both current position and direction of each airplane, see Figure 14. Different types of airplanes are differentiated by the shape of their icons.

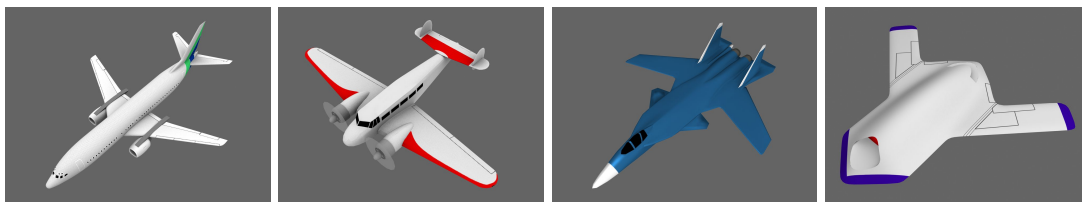


Figure 14: Four types of airplanes: Boeing 737, Electra and Su-47 and X-45

Recent flight history of each airplane is visualized by a polyline of the same color as the color of the airplane's icon. It can be switched on and off by pressing the H key.



An airplane can be selected by clicking on it. This applies to both 2D and 3D mode. It is a common practice to select the airplane in 2D mode and then switch to 3D mode rather than to try selecting the airplane directly in 3D, which can be tricky sometimes. Once the airplane is selected, the camera starts following it so that it stays in focus, see Figure 15. A set of zones is displayed around the selected airplane as a group of concentric circles. Most importantly, these refer to the collision range (innermost) and the visibility range (outermost) of each airplane. Once an airplane enters the visibility range of the currently selected airplane, its icon changes to a bright (and preferably unique) color while all icons of airplanes that fall outside the visibility range are colored in dark grey. This makes it easy to quickly identify and track the airplanes that are visible to the currently selected airplane.

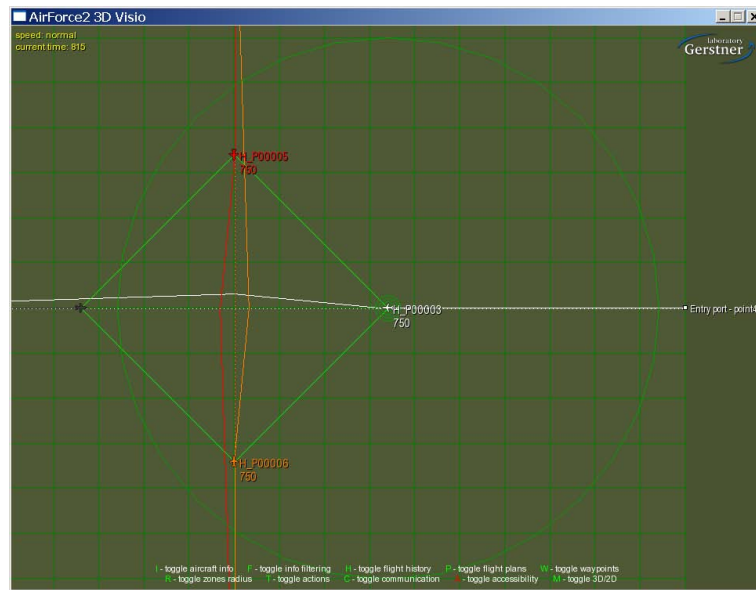


Figure 15: Communication lines between airplanes

Communication between a pair of airplanes (which can occur only in case that they both fall into the visibility range of one another) is represented by a green solid line connecting the two airplanes. Visualization of communication lines can be switched on and off by pressing the C key.

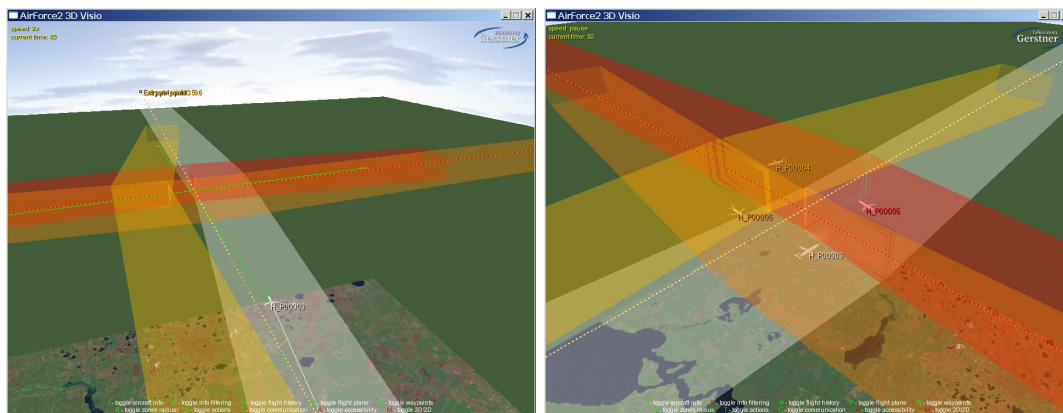


Figure 16: Flight plans in 3D



The current flight plan can be displayed for the selected airplane and/or for all airplanes that are visible to it. The user can circle through these various modes by pressing the P key. Flight plans in 2D are represented by solid polylines that interpolate through the intended route (Figure 15). In 3D, flight plans are displayed as three-dimensional semitransparent corridors of a rectangular profile, see Figure 16. Once again, the color of each flight plan matches the color of the airplane's icon.

Waypoints of the currently selected airplane in both 2D and 3D mode are displayed as a set of vertices in 2D/3D space and are interpolated by dotted polylines. If no airplane is selected, then all the waypoint vertices are displayed and by clicking on any one of them, the entire corresponding interpolated path is rendered. Waypoints can be switched on and off altogether by pressing the W key.

The 3D mode is useful for observing the entire simulation in a natural 3D environment that allows the user to get a better insight of the spatial relations between airplanes, the actual 3D shapes and possible collisions of flight plans, series of waypoints etc. The 3D mode provides almost all the information earlier described in case of 2D mode.

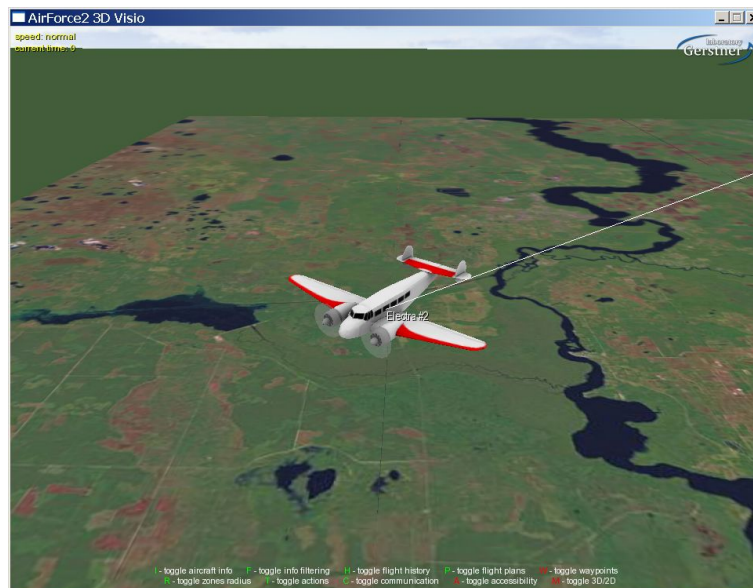


Figure 17: 3D view

Apart from other things, in 3D mode it is possible to hover around the selected airplane by holding the SHIFT key and pressing the arrow keys. It is also possible to zoom in and out by pressing UP and DOWN arrow keys alone. These two features are particularly useful when trying to get the most convenient view of the current state of the simulation.

Another important feature of both the 2D and 3D visio is the ability to interactively control the speed of the simulation. The user can achieve this by pressing the following keys: NUMPAD PLUS (increase the speed), NUMPAD MINUS (decrease the speed) and NUMPAD STAR (toggle pause on/off). As the user presses any of these keys, the visio sends a message to the simulation system to change the speed of the simulation and the simulation system subsequently sends a confirmation message back to the visio with the information about the newly set simulation speed. The visio displays the current speed of the simulation in the top left-hand corner of the screen, together with the current simulation time.

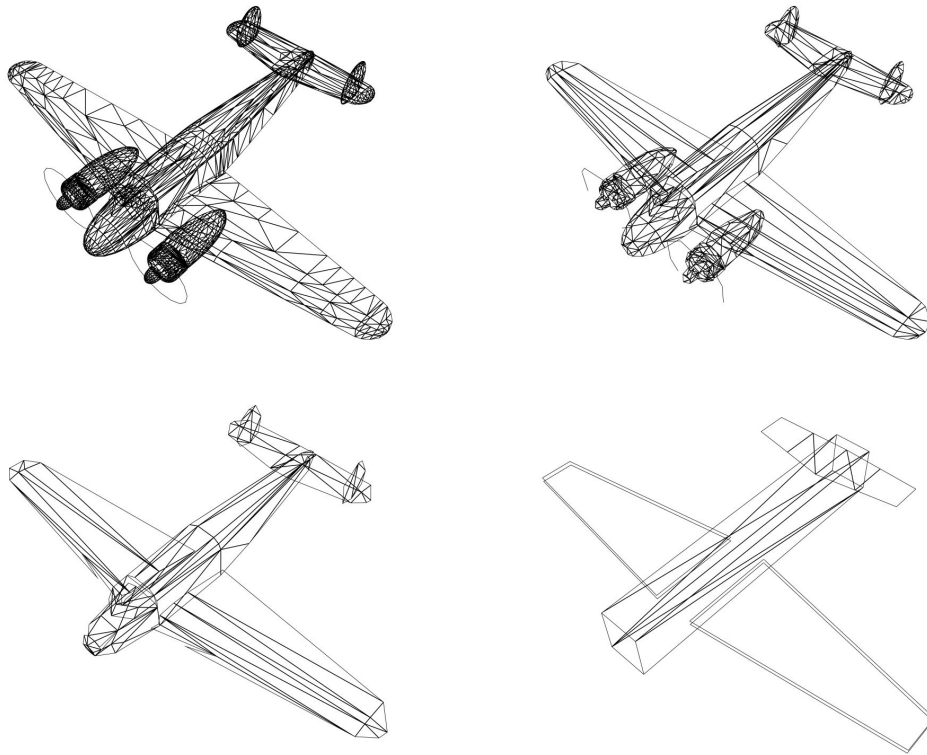


Figure 18: Multiple levels of detail of the same 3D model

As the visio is required to cope with as many as hundreds of airplanes, massive optimization of visualization of 3D space had to happen. One of the key features in this respect is the incorporation of multiple levels of detail (LOD) for all 3D models (Figure 18). For closeup views, the 3D models contain as many details as possible in order to look realistic. But as the camera moves further away, the detailed models are replaced with more rough and therefore more lightweight versions. For wide shots, the 3D models consist only of a few polygons and so they can be rendered very fast. Each airplane exists in five LOD representations ranging from tens to several thousands of polygons.

5.3 Web based Access to the Application

The simulation can also be accessed using a Java client web application that via network connects to the simulation system which acts as a server and provides all its clients with regular data updates. This way, a number of users can concurrently observe and interact with the very same simulation. Accessing the simulation system via network is as simple and straightforward as opening a web browser and entering the IP address of the computer on which the **A-globe** simulation system is currently running. The client web application will launch shortly.

Before the user can access the simulation, he/she needs to log in the system by entering a valid user name and password as shown in Figure 19.

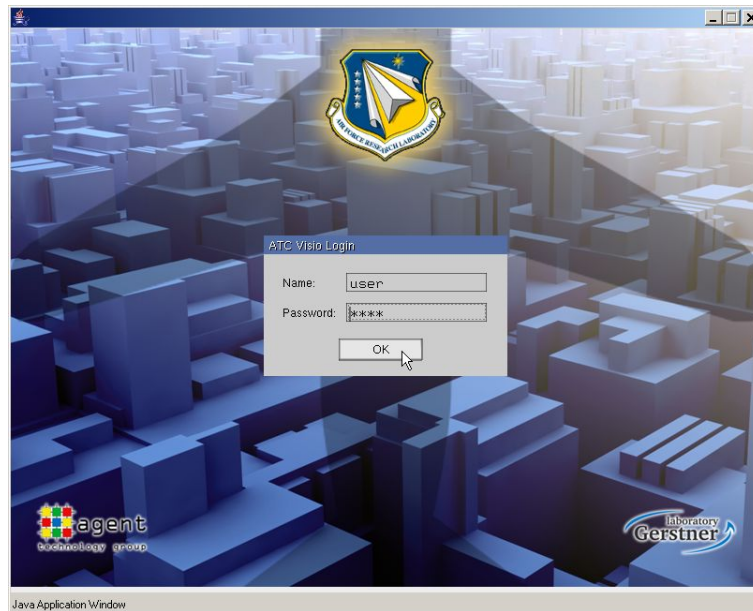


Figure 19: User login

5.3.1 GIS Data Layers

During the development of the simulation system, a request arose that the simulation should be based on and integrated with real world data. Therefore it was necessary to start looking for internet resources of various GIS data. Fortunately enough, it quickly turned out that such data are publicly available online, at least for the area of the United States. Data from several websites were parsed, filtered and merged together to serve as a database for our simulation system.

- **Landsat7 Images** – a mosaic of Landsat7 images at the maximum resolution of approximately 50 meters per pixel was used as an underlying orthophoto map of the United States. The source data for this layer were collected from <http://onearth.jpl.nasa.gov>. For purposes of performance optimization and data flow reduction, the original satellite images had to be split into uniform tiles of 512×512 pixels. The same image mosaic was generated in multiple resolutions to allow seamless zooming in and out of the virtual map. Since the data for this layer consist of about 1.7 gigabytes of JPEG-compressed images covering the entire area of the United States, a sophisticated system of switching between image tiles and multiple resolution representations was the only and the best way to manage such amount of data. See Figure 21.
- **State Boundaries** – detailed vector shapes of 50 U. S. state boundaries obtained from <http://seamless.usgs.gov>. Extensive post-processing of the data had to be carried out as the source data contained a lot of shape redundancies and duplicities. See Figure 21.
- A** **Airports** – a set of more than 650 U. S. airports, including their names, GPS coordinates and the corresponding average numbers of enplanements per year obtained from the site <http://seamless.usgs.gov>. The size of the airport icon reflects the average number of enplanements per year. See Figure 22.
- **No-flight Zones** – for the purposes of the simulation we decided to use U. S. powerplants to act as the no-flight zones. A set of more than 80 U. S. powerplants, including their names

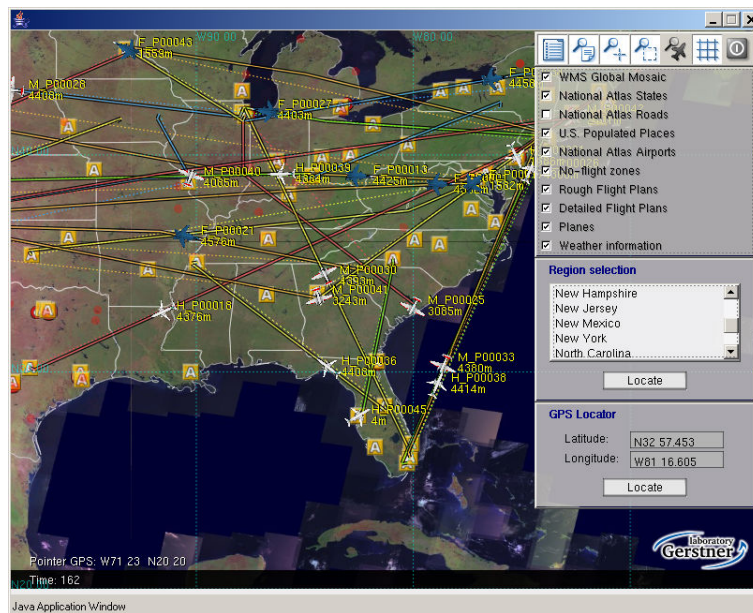




Figure 20: User interface overview

and GPS coordinates was derived from the data available at <http://geonames.usgs.gov>. See Figure 22.

 **Cities** – a set of more than 24 thousand U. S. populated places, including their names, GPS coordinates and the corresponding population obtained from <http://geonames.usgs.gov>. The size of the city icon reflects the size of the population. See Figure 23.

 **Highways** – a set of more than 26 thousand major U. S. highway segments derived from the data available at <http://seamless.usgs.gov>. See Figure 23.

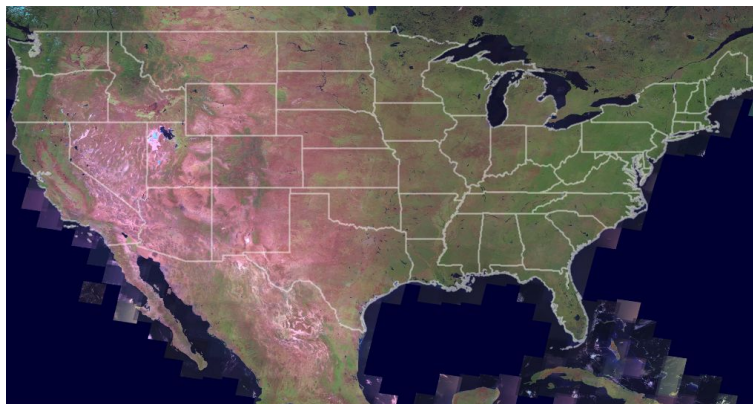


Figure 21: GIS data layers: Landsat7 images and U. S. state boundaries

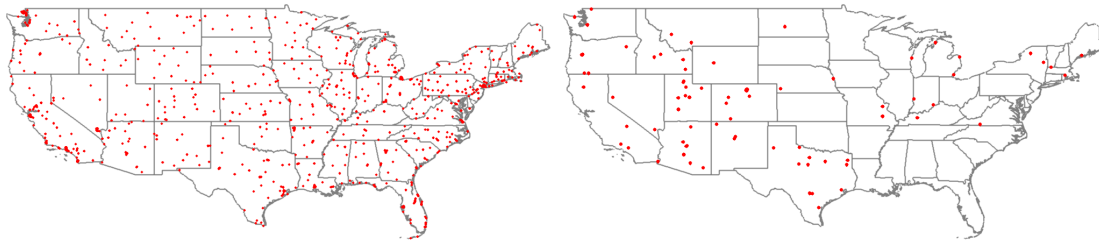


Figure 22: GIS data layers: airports (left) and no-flight zones (right)

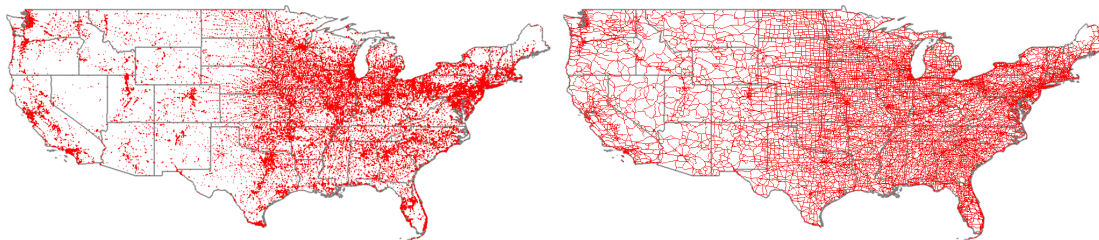


Figure 23: GIS data layers: cities (left) and highways (right)

5.3.2 ATC System Data Layers

Apart from the generally static GIS data layers, there are also several ATC data layers that are generated and updated by the simulation system in runtime.



Airplanes – a layer of airplane icons representing each airplane's type, current position and direction. See Figures 20 and 25.



Rough Flight Plans – also called waypoints, these plans are series of points in space through which the airplane is supposed to fly on its way. These are displayed as dotted polylines in the visio. See Figure 20.



Detailed Flight Plans – these plans are the actual routes that the airplanes follow. These are displayed as solid polylines in the visio. See Figures 20 and 25.

- **Weather Information** – weather conditions in the local area are displayed in the status line at the bottom of the screen. See Figures 24 and 25.

5.3.3 Icon Palette and Keyboard Controls

In the top right hand corner of the screen is located an icon palette which gives the user access to application controls.



Layer Menu – lets the user switch on and off various data layers in the main view

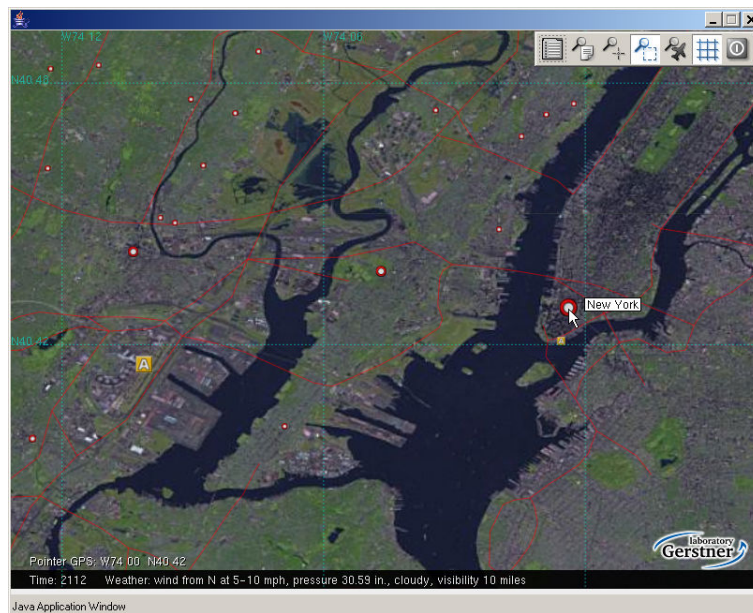








Figure 24: Detail view including the weather information

-  **Zoom to Region** – lets the user zoom to a specific region (state) by selecting it from the list
-  **Locate GPS** – lets the user move to a specific location by entering its GPS coordinates
-  **Zoom to Selection** – lets the user zoom to a specific part of the scene by selecting a rectangular area with the mouse
-  **Zoom to Airplane** – lets the user zoom to a specific airplane by clicking on it
-  **Toggle Coordinate Grid** – lets the user toggle the GPS coordinate grid
-  **Logout** – lets the user log out from the system

The user can also use the keyboard to navigate in the main view:

- **arrow keys** – scroll/pan the view
- **PageUp, PageDown** – zoom the view in and out

As the user changes the view, the appropriate data segment of the virtual map is requested and downloaded from the server, and therefore at any moment the client needs to keep only a minimal amount of data, which results in its fast and efficient performance.

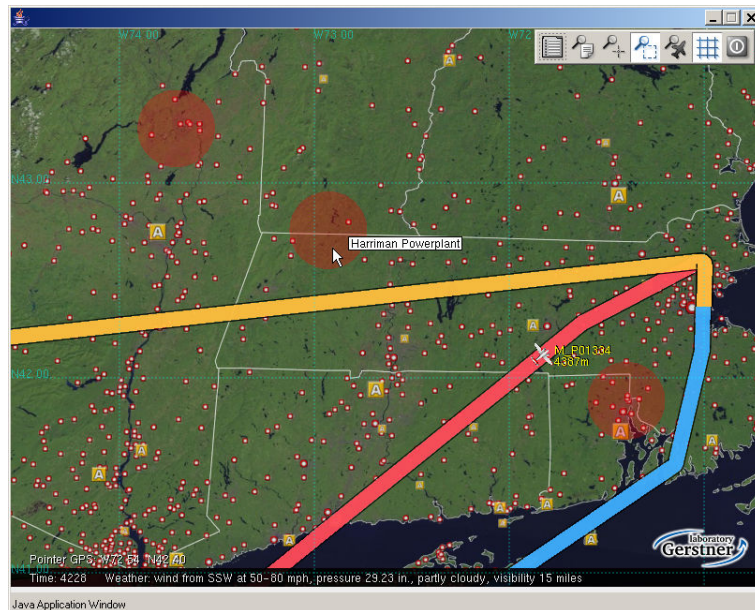


Figure 25: Flight plans avoiding no-flight zones



6 Negotiation based Deconfliction

The designed ATC core system plans deconflicted flight routes entirely in a distributed manner. There is no central planner managing all flight plans for the simulated area. Therefore the deconfliction technology (while developed with in the multi-agent ATC model) is ready for deployment on autonomous vehicles without any central point of control.

Simulated airplanes are independent agents. Each agent representing an aircraft auto-pilot is a self-interested entity which prepares a detailed flight plan for the airplane with respect to waypoints specified in the airplane's mission. Each simulated airplane is surrounded by a number of concentric spherical zones, see Figure 26 (sorted by its radius):

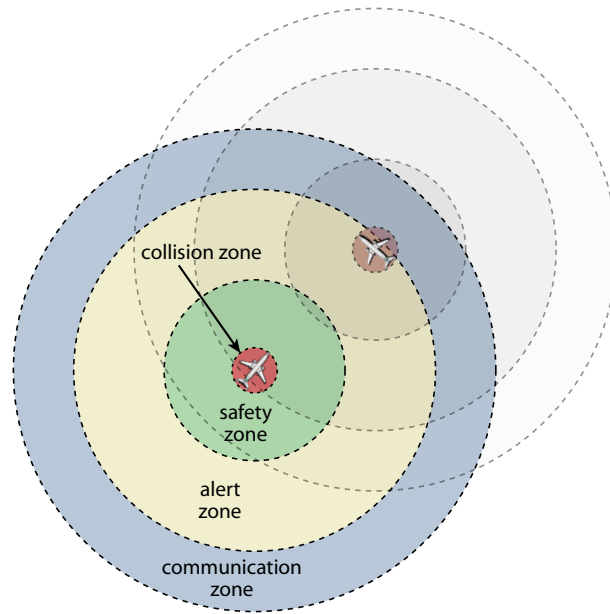


Figure 26: Communication, Alert, Safety and Collision Zones around each aircraft

- **Communication zone** – is the outermost one. It represents the communication range of the data transmitter on the aircraft board. Using this data, transmitter plane agents can send data packets to other planes which are located within the specified spherical zone defined by its radius. Two airplanes can communicate in both directions if the distance between them is smaller than the maximum of their communication radius.
- **Alert zone** – defines the operation range of the radar on the aircraft board. If another airplane is located within the zone, plane agents are periodically notified about its relative position and its flight code; this situation is shown in Figure 26.
- **Safety zone** – encapsulates the area around an airplane that other airplanes should not enter in order to minimize the mutual influence of the airplane movements, e.g. airspace turbulences. Two airplanes should not come closer than the maximum of their safety ranges. If it happens, airplanes can still continue flying but their flight path may be influenced by turbulences etc. This is not the case when two or more airplanes fly together in a close formation.



- **Collision zone** – is the innermost zone. It defines the critical contact area. When two airplanes get too close together and their mutual distance is smaller than the sum of their collision radiuses, physical contact between them occurs.

Zone ranges are the same for all airplanes of a certain type. Their sizes can be defined independently for each of the airplane types.

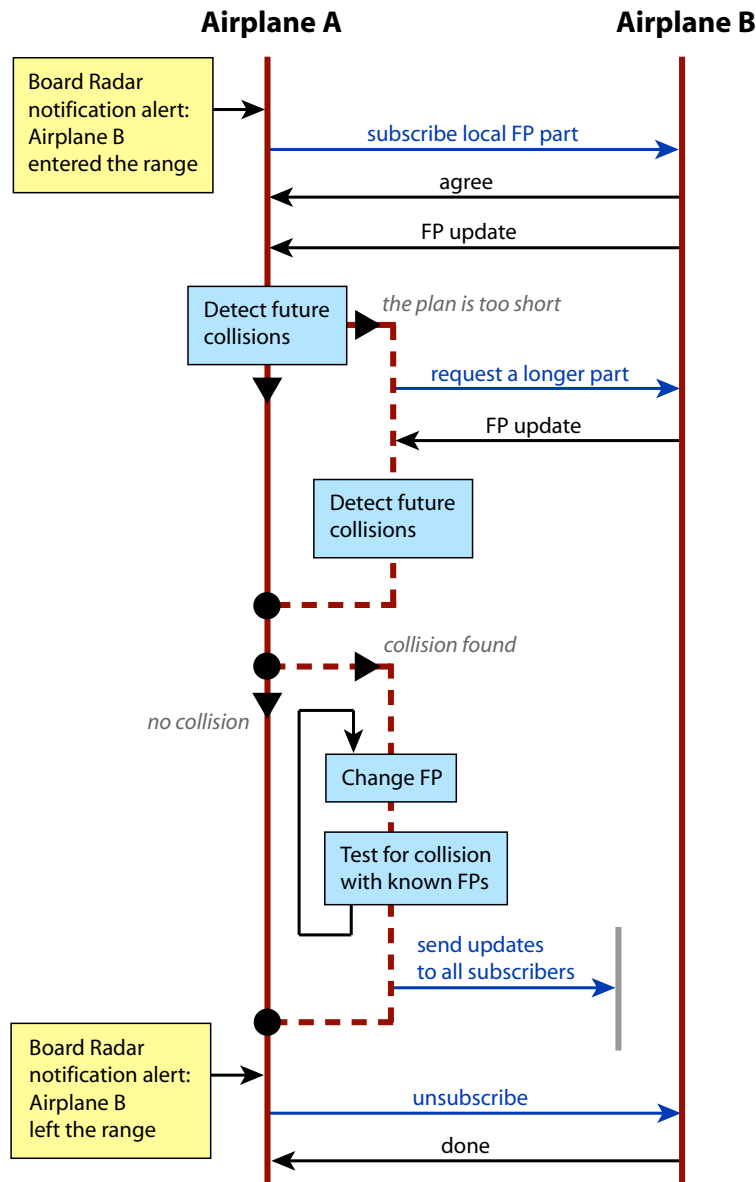


Figure 27: Negotiation protocol

Currently the ATC system solves collisions cooperatively by negotiation between airplanes, see Figure 27. Let's suppose an airplane A is flying along its planned optimal flight path through its mission waypoints. Airplane B enters the *alert zone* of airplane A. The pilot agent of the airplane A is notified about its position and flight code by the board radar system. The pilot agent A



tries to establish negotiation connection with the pilot agent B. In case the connection cannot be established or the communication is not trusted, the pilot agents should use *non-cooperative approach*, described later in this section. If the connection was established successfully, the pilot agent A subscribes for a local area flight plan of the airplane B. The pilot agent of airplane B sends an update to the subscriber every time it changes its own flight plan. The update contains the airplane's flight plan part for the specified amount of time depending on the flight speed. The update is also sent when the time length of the previous update was not long enough and it needs to be updated again. When the pilot agent A receives an update from the pilot agent B, it executes the collision detection procedure described in section 6.1 on its own flight plan and the received one. If the detection test is negative, both flight paths are safe for flying. If a collision is found, the airplanes A and B must change their flight paths.

Currently in the ATC system there is implemented a rule-based approach for changing flight plans described in section 6.2. The system will be extended so that airplanes that detected future collisions will iterate through *monotonic concession protocol* to find new flight plans that are collision free. Both airplanes prepare a set of possible flight plan changes scored by the utility function. The utility function includes pilot's own intentions including flight priority, fuel restrictions, time restrictions, etc. From all collision free combinations of flight plan pairs, the possible solution set is created. The iteration protocol results in a commonly accepted solution of the collision. Then each airplane applies the respective flight plan changes. The iteration solution has many advantages compared to the rule-based approach. The rule-base approach assumes that all airplanes use the same deconfliction rules. However, the iteration solution can lead to the situation when the solution is not found fast enough. The process has to be extended with an emergency solution that is used when the iteration process doesn't lead to any fast solution. As the emergency solution, the game theory approach can be used. This will be described in the following paragraph.

The distributed deconfliction approach can be easily extended to *non-cooperative deconfliction*. The non-cooperative deconfliction is useful in situations when an airplane has a malfunctioning transmitter/receiver on its board or in the situation when an airplane is an intruder/enemy which intentionally sends incorrect future flight path parts to others. The most suitable approach to the *non-cooperative deconfliction* is the game theory. In this case the pilot agent tries to change its own flight plan in a way that would guarantee a minimal collision risk for any future position of the other airplane. To determine all possible future positions of the other plane, information about its current position, direction and information about its type can be used. The monitored object's flight path is always continuous but there are also certain restrictions that depend of the airplane type – e.g. minimal/maximal flight speed, minimal radius of turning, etc. When the pilot agent wants to identify whether or not it should use the non-cooperative deconfliction for a particular airplane, it can integrate a special *detection module*. The detection module compares the notification received from the board radar with the known flight plan part of the aircraft in the radar range.

6.1 Collision Detection

Integrated collision detector is described in this section. Current detector works with two flight plans (representation of flight plan is described in the section 4.2). Each flight path contains time information. The detector goes along both flight paths in the time and tests the condition of safe flights described in previous section. If time instant is found when the distance between airplane positions is smaller than maximum of the planes safety ranges, the collision detection process is positive, see Figure 28. By this way the detector identifies **time 1** which represents first collision point on the plane flight plan.

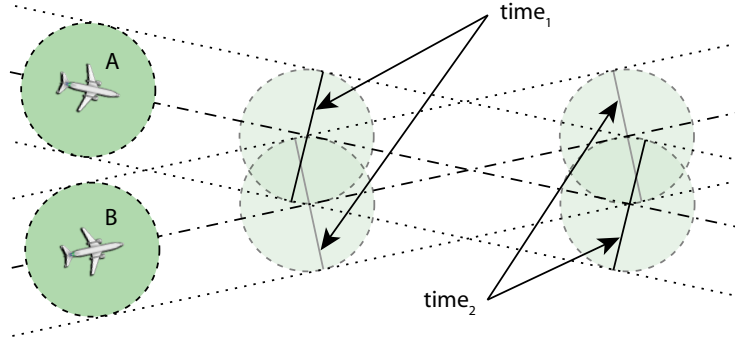


Figure 28: Flight plan collision interval

The pilot agent also needs to know the length of the flight plan collision interval for better situation handling. Detector thus also seeks for the **time 2** which represents the last collision point between the flight plans. If the flight plan part provided by the second aircraft ends before the last collision point is found, the pilot agent re-requests next part of the other plane's flight plan until the last collision point is found.

6.2 Rule-based Approach

A rule-based approach for flight plans deconfliction to no-collision pair is described in this section. Type of the collision between the airplanes is identified first. The collision type is determined on basis of angle between direction vectors of the concerned planes at **time 1** projected to the ground plane (defined by X and Y axis), see figure 29.

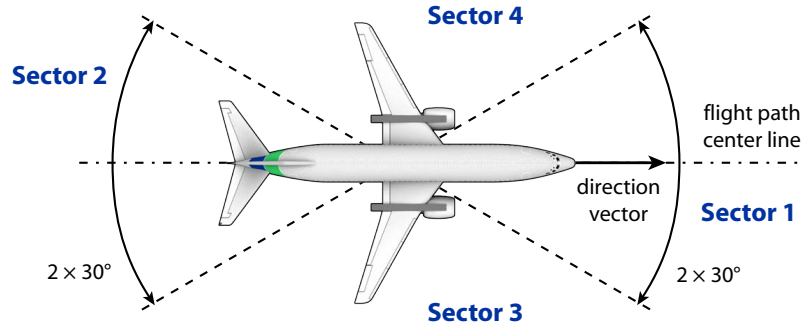


Figure 29: Identification of the collision type

Depending on the computed angle the plane B is relatively to the plane A it fits into one of the four sectors. Depending on this sector, one of the following rules is applied on the flight plan of plane A to avoid the collision:

- **Sector 1** – head-on collision, in this case the planes avoids each other on the right side of the second one. The plane flight plan is changed as shown in Figure 30. The pilot agent



shifts the points in the **time 1** and **time 2** perpendicularly to the old direction vector to the right. Distance between the previous and new points is equal to minimum of safety ranges. After **time 2** flight plan will continue shortest way to the next mission waypoint.

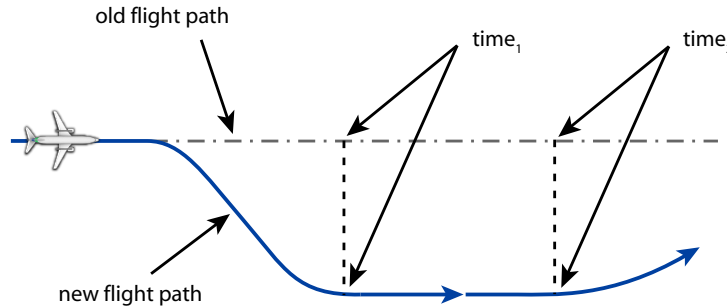


Figure 30: The change of the flight plan applied in the case of head-on collision

- **Sector 2** – back collision, there are two subcases: i) the aircraft which is in front of the second one is faster – aircrafts do not change current flight plans. ii) The back and faster plane changes its flight plan so it will pass the front plane on the right side without endangering the front one. The flight plan is similar to that in Figure 30. Again the back plane shifts the old points in the **time 1** and **time 2** perpendicularly to the old direction vector to the right at the distance at least 1.1 times of the safety range.
- **Sector 3** – side collision when the other plane has traffic priority. The aircraft needs to slow down its speed so that it reaches the first collision point later than the second airplane. If this is not possible due to the minimal flight speed defined for each plane type, the aircraft slows down as much as possible and shifts the flight point from the first collision point to the right so that there is no collision between their flight plans.
- **Sector 4** – side collision when this plane has traffic priority. The aircraft changes its flight plan by accelerating up to its maximal flight speed so that it passes the collision point before the other airplane. The plane only accelerates as much as needed.

The above rule-based changes to the flight plan are done by both planes independently because the second aircraft detects the possible collision with the first plane from its point of view. After applying the changes to the aircraft flight plan, it sends an updated local flight plan part to all subscribers (planes located in its vicinity). The change is also verified against all other known flight plans of all aircraft monitored by the board radar system. If there is another collision detected, new changes are applied.

The pilot agent internally uses the *flight plan wrapper* interface for manipulation with its flight plan. The change requests are handled as a special set of solver time-constrained waypoints. Special handling algorithm implements the application of a new change that overrides the old one. The algorithm decides whether an older solver waypoint should be removed or not.

6.3 Planning Deconflicted Trajectories

The designed ATC system as described does not produce a set of deconflicted route plans for the aerial vehicles. The novelty and innovation in the approach is that the planning system (during the



planning phase) provides the flight plans that can contain possible collisions. This makes planning considerably faster and much more flexible. This results in substantial scalability improvement affording to operate higher number of aerial vehicles in condensed airspace (can be used e.g. in situation where a higher number of unmanned aerial vehicles carries out rapid surveillance tasks).

Even though that the flight plans are straightforward and can include possible collisions, the deconfliction mechanism makes sure that during the flight (simulated in our application by the *simulation phase*) the pilot agents negotiate rational deconfliction process. Scalability of the deconfliction task will be tested in the next phase of the project.

However, the ATC system can be equally used for centralised planning of **deconflicted flight plans**. The planning operation of the system can result in a set of plans that are collision free.

This functionality is achieved by adding yet another phase in the operation of the system – *post-planning deconfliction phase/mode*. In this phase or better denoted as an operation mode the simulation process containing the deconfliction processes (as described in 6) is performed. All resource consuming visualisation operations are disabled and the operation of the aerial vehicles planes is simulated in very fast manner.

During the post-planning deconfliction phase the collision free flight plans of the simulated aircraft are remembered. These flight plans are then used in the simulation phase so there is no other collision. In this situation this phase works as a verification phase.

The ATC system thus would operate in three phases:

1. planning
2. post-planning deconfliction
3. deconflicted plan simulation

Use of ATC system for planning deconflicted airways has only sense in situations when we know all flight details of all aircraft in advance, all simulated planes cooperatively solve collisions and they are truthful. When there is at least one non-cooperative/intruder/enemy aircraft, this *pre-planning phase* has no sense because we cannot know what will non-cooperative aircraft do in advance. In this case it is better to use ATC system for re-planning flight plan for solving possible conflicts just-in-time. The pilot agent can utilize periodical information from the board radar equipment for monitoring the other plane movement. The agent tries to change only its own flight plan without any communication based on game theory approach.



7 Demonstration Disc

The Air Traffic System (ATC) demonstration disc is included in this project deliverable. To use the demonstration disc the standard computer with DVD mechanics is required. As all scripts are prepared for the Microsoft Windows 32 bit operating system, we suggest to use Windows 2000/XP. The JAVA based part of the system can run on any operating system supporting Java Runtime Environment. The disc contains:

- two demonstration scenarios – first demonstrates integration of the ATC system with external data sources (see section 5.3.1) and provides remote WEB client access to the system. In this demonstration, the external content is stored on the disc. Second configuration demonstrates the mechanism of distributed deconfliction between four airplanes converging to the same point. In the second scenario the real-time 2D/3D visualizer is used (see section 5.2),
- technical documentation – contains detailed description of classes and properties, this report and the demo starting procedure,
- sources – all JAVA sources of the system with comments. **A-globe** sources are included,
- video – demonstration video is provided to show the key features of the project when the full prototype can't be started. The video is encoded with the DivX 6.0.3 video codec. Installation package for the coded is included.

7.1 System Requirements: System Core

One or more host computers with following requirements are required to run the ATC Core. Use of several computers allows higher number of deconflicted airways to be planned because planning is handled in a distributed manner based on plane-to-plane negotiation.

- JAVA supported operating system with 512MB RAM at least and TCP/IP protocol support,
- Java Runtime Environment 1.5 or higher (version for Windows 32bit is pre-installed on the distribution disk),
- network connection between all computers running Agents Core System (not necessary if whole system is running on the one host) **without any firewall restrictions**.

7.2 System Requirements: Remote WEB Client

- standard PC with network connection to the System Core host (intranet/internet),
- Windows 32bit, Linux, Mac OS X, Sun OS (sparc or x86), 512 MB RAM at least,
- graphics drivers with OpenGL support,
- pre-installed Java Web Start (Java Runtime Environment 1.5 or higher is optional because it can be downloaded and installed by Java Web Start automatically),
- any internet browser associating the Java Web Start application with JNLP extension.



7.3 System Requirements: Real-time 2D/3D Visualizer

- Windows 32bit operating system, CPU 1GHz+, RAM 512MB+ and TCP/IP protocol support,
- graphic card with hardware 2D/3D acceleration and OpenGL support in the drivers,
- minimal screen resolution 800x600 pixels,
- network connection to the Core System host (only necessary when real-time visualizer is running on different host).



References

- [1] A-globe. A-globe Agent Platform. <http://agents.felk.cvut.cz/aglobe>, 2005.
- [2] CrystalSpace. Crystal Space 3D - opensource game engine. <http://www.crystalspace3d.org>, 2004.
- [3] JOGL. Java Bindings for OpenGL. <http://jogl.dev.java.net>, 2005.
- [4] David Šišlák, Martin Rehák, Michal Pěchouček, Milan Rollo, and Dušan Pavlíček. **A-globe**: Agent development platform with inaccessibility and mobility support. In Rainer Unland, Matthias Klusch, and Monique Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.